Faculty of Science and Engineering
Department of Mathematics and Statistics

Optimisation Techniques for Natural Gas Distribution Networks

Efat Fakhar

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

March 2018

Declaration

To the best of my knowledge and belief, this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

.....
Efat Fakhar
5/03/2018

Acknowledgements

I wish to express my sincere thanks to my supervisors, Professor Louis Caccetta and Doctor Elham Mardaneh for their continuous support, guidance, and encouragement.

I would like to thank my beloved, Rahim my husband for his support, patience and encouragement, and Armita my daughter for being understanding, encouraging, and cheering me up.

I would also like to indicate my thanks to Curtin University for providing me with financial support (Curtin University Postgraduate Scholarship (CUPS)) and making this journey possible.

Abstract

The process of moving natural gas from the gas fields to consumers is complex and requires the resolution of a number of problems from the extraction to the processing and distribution of products. The methods of Operations Research have been successfully applied to a number of problems arising in each of the four main stages of the gas industry namely: natural gas production, transmission, distribution, and marketing. Many researchers have worked on finding optimal solutions in different areas of the natural gas industry to minimise the cost of networks and to increase customer satisfaction. Some details of these is given in Chapter 1.

There has been very little work on the natural gas distribution area in the literature. In this thesis, we consider the network design and allocation problem for natural gas distribution networks. A detailed literature review is presented in Chapter 2. Our fundamental problem is, given a set of nodes and customer requirements, determine the optimal network. This includes the network layout, the diameter type for the connected links, the pressure at each node, and the flow through each link. Our aim is to develop an effective mathematical model for this fundamental problem. The objective is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which in our case is a tree.

There are three cases of pipeline diameters to consider: single diameter; continuous diameter; and multi diameter. In the literature, the cases of single diameter and continuous diameter have been considered by Rothfarb, Frank, Rosenbaum, Steiglitz, and Kleitman (1970) and Andre, Auray, Brac, De Wolf, Maisonnier, Ould-Sidi, and Simonnet (2013). There is no work in the literature that

considers the case of multi diameter. The contributions of our research are in the area of gas distribution network design and allocation that have unresolved issues. More specifically we develop an effective Mixed Integer Non Linear Programming (MINLP) model for the fundamental problem in the case of multi diameter. The multi diameter case corresponds to where a link consists of different commercially available diameters that are serially connected. The objective function is a linear cost function of the pipe diameters. We consider a wide range of design parameters including the number of demand nodes, the set of pipe diameter types, the length (distance) between nodes, and the pressure limits at each node. We consider the constraints under steady-state conditions such as pressure limits at each node, the flow balance through each link, the pressure drop equation for two ends of each link. The decision variables are the selected links connecting nodes, the flow through each link, the pressure at each node, and the length of selected diameter for each link. Our model includes a linear objective function and the nonlinear and nonconvex equality constraints. These constraints make our MINLP model computationally difficult. We develop two different solution methods for our model. One is an approximation method and the other a heuristic method.

We are motivated to solve our MINLP model using a method that handles the nonconvexities of the constraints. The algorithm that we use is the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method given by Viswanathan and Grossmann (1990). Our motivation to choose this OA/ER/AP method among the other MINLP algorithms, is in its ability to handle the nonconvexity of the problem. This algorithm alternates between two sub-problems including a Non Linear Programming (NLP) sub-problem and the Master Mixed Integer Linear Programming (MILP) sub-problem. In the NLP sub-problem, the integer variables are fixed and the continuous variables are determined. In the master MILP sub-problem, the effect of nonconvexities are reduced by linearization of the nonlinear functions at the set of linearization points. Our contribution is that we review this algorithm and then examine its functionality and efficiency on our MINLP model. We consider the effects of different parameters and stopping criteria in the NLP sub-problem and the Master MILP sub-problem to find a near optimal solution. The main outcome is a MINLP model that is effective for small sized networks. The details

are given in Chapter 3.

The computational results for the OA/ER/AP algorithm are undertaken in Chapter 4 considering different values of optimality gap for the Master MILP sub-problem. We acknowledge the single starting point and the multi starting points for the NLP sub-problems. Also, we discuss the comparative analysis between the NLP-Single start and the NLP-Multi start strategies considering different values of optimality gap for the Master MILP sub-problem. For some test cases, a decrease in the value of ROT(MILP) results in an increase in the computation time and an improvement in the total cost. As expected the NLP-Multi start yields an improved objective function value but requires more computational time than the NLP-Single start.

We present a new heuristic algorithm to solve our MINLP model in Chapter 5. Our model is computationally difficult to solve using exact methods in particular for large sized test problems (50 customers). Our motivation is to develop a heuristic algorithm to solve large size networks. We allow our heuristic 4 hours of computation time so that it generates a good quality for the large networks in the test cases. The developed algorithm includes two levels: in the outer level, a tree network is generated. Then, in the inner level, we determine the multi diameters for the given links as well as determining the pressure at each node and the flow through each selected link by solving the remaining linear sub-problem to find a cost-effective solution. The detailed algorithm is presented, and different size numerical examples are used to test the efficiency of the algorithm. The quantitative analysis of the effects of different parameters of our heuristic algorithm on optimal decisions is also investigated. Also, we compare the results in the objective function value and the computation time between the OA/ER/AP algorithm and our heuristic algorithm. The main outcome is a heuristic algorithm that finds a good quality feasible solution for the large sized test problems within 4 hours.

In Chapter 6, we provide a summary of the earlier chapters and conclusions obtained from the research. Also, we present possible future work in areas that are open to further investigation and perhaps improvements.

Dedication

This thesis is dedicated to my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural gas has significant environmental benefits over both coal and oil. It provides 65 to 70 percent less greenhouse gas and other emissions than fossil fuels. Natural gas is used in large quantities by commercial and residential users as well as the industrial and electrical sectors. Also, with the increasing concern of climate change and the subsequent regulations such as using low carbon fuels, gas consumptions will increase, especially for electricity generation. Besides, gas can be used as an alternative fuel for transportation in the form of Compressed Natural Gas (CNG) or Liquefied Natural Gas (LNG) (Roarty and Roarty (2008)).

Operations Research has played a significant role in the natural gas industry to solve a number of essential and relevant problems in the areas of production, transmission, distribution, and marketing. Indeed, during the past 45 years, many issues in the gas industry have been tackled by operations research models and techniques. Many researchers have worked on finding optimal solutions in different areas of the natural gas industry to minimise the cost of networks and to increase customer satisfaction. However, very little of this research is in the gas distribution area. In this chapter, we present an overview of natural gas including the composition, the formation, and the uses of natural gas. Also, we present an overview of the gas industry supply chain as well as the number of problems arising in all four of the above-mentioned areas of the gas industry. We focus on, in particular, Gas Distribution Networks.

This chapter is organised as follows: Section 1.1 gives an introduction to the composition, the formation, and the uses of natural gas. The gas industry supply

chain, including all sections to move natural gas from the gas field to consumers, is presented in Section 1.2. In Section 1.3, we present a number of problems arising in each of the four main stages of the gas industry production, transmission, distribution, and marketing. Section 1.4 provides a brief outline of this thesis.

1.1 Overview of Natural Gas

In this section, we provide the composition, the formation, and the uses of natural gas in more detail.

1.1.1 Composition of Natural Gas

Natural gas is a combustible mixture of hydrocarbon gases. While natural gas is formed primarily of methane, it can also include Ethane, Propane, Butane and Pentane. The composition of natural gas can vary widely, but below is a chart outlining the typical make up of natural gas before it is refined (NatGas (2013)).

Table 1.1: Typical composition of natural gas (NatGas (2013))

Gas name	Symbol	Value
Methane	CH4	70-90%
Ethane	C2H6	0-20%
Propane	C3H8	
Butane	C4H10	
Carbon Dioxide	CO2	0-8%
Oxygen	O2	0-0.2%
Nitrogen	N2	0-5%
Hydrogen sulphide	H2S	0-5%
Rare gases	A, He, Ne, Xe	trace

However, the delivered natural gas to consumer's home is almost pure methane. The distinctive "rotten egg" smell that we often associate with natural gas is an Odourant called Mercaptan that is added to the gas before it is delivered to the end user. Mercaptan aids in detecting any leaks (NatGas (2013)).

1.1.2 The Formation of Natural Gas

Natural gas is formed from the decomposed remains of plant and organic material. The decomposition over millions of years creates gases that become trapped in different rock formations, including sandstone, tight sandstone, shale, and coal seams (NatGas (2013)). These rock formations are outlined in more detail below:

- **Coal Seam Gas:** Coal seam gas is trapped in coal formations. The coal is a sedimentary rock created from the compressed remains of organic material such as plants when the heat and pressure increased. The coal seams are generally filled with water, and it is the pressure of the water that keeps the gas as a thin film on the surface of the coal.

- **Shale Gas:** Shales are fine-grained sedimentary rocks, formed from the compaction of silt and mud.

- **Tight Gas:** Tight rocks are typically limestone and sandstone. Both shale and tight rocks have deficient levels of permeability and are found deep underground.

As shown in Table 1.1, natural gas is an odourless, colourless mixture of gases containing mainly of methane.

1.1.3 Uses of Natural Gas

Natural gas has a wide range of applications and can be used in some sectors as detailed below (NatGas (2013)):

1. **Residential:** The best-known uses for natural gas around the home are natural gas heating and cooking.

2. **Commercial:** The commercial sector includes public and private enterprises such as office buildings, schools, churches, hotels, restaurants, and government buildings. The main uses of natural gas in this sector include space heating, water heating, and cooling that are similar to residential uses.

3. **Industrial:** Industrial applications include those same uses of residential and commercial such as heating, cooling, and cooking. Natural gas is also utilised for waste treatment and incineration, metals preheating (particularly for iron and steel), drying and dehumidification, glass melting, food processing, and fuelling industrial boilers. The natural gas may also be used as a feedstock for the manufacturing of a number of chemicals and products. Gases such as butane, ethane, and propane may be extracted from the natural gas to be employed as a feedstock for such products as

fertilisers and pharmaceutical products (NatGas (2013)).

4. **Electricity generation:** Natural gas can be used to generate electricity in a variety of ways (NatGas (2013)).
   - Steam generation units: The most basic natural gas-fired electricity generation consists of a steam generation unit, where gas is burned in a boiler to heat water and produce steam that then turns a turbine to generate electricity.
   - Centralised gas turbines: Gas turbines and combustion engines are also used to generate electricity. In these types of units, instead of heating steam to turn a turbine, hot gases from burning natural gas are used to turn the turbine and generate electricity.
   - Combined cycle units: These units use both a gas turbine and a steam unit, with the waste heat from the gas-turbine process used to generate steam.
   - Natural gas can also be used to power back-up generators and distribute generated energy.

5. **Transportation:** Natural gas is an available alternative fuel particularly useful in the transportation sector. As natural gas is low emission, it makes it easier for new vehicles to meet increasing environmental standards.

1.2 Natural Gas Supply Chain

The process of moving natural gas from the gas field to consumers is detailed in Figure 1.1 and includes the following stages. We need to note that we have permission from Stevens (2012) to use this figure in our thesis. Also, this image has been downloaded from https://barryonenergy.wordpress.com.

1. **Exploration:** To identify the potential source of natural gas. Extensive studies are undertaken to assess the viability of the gas reserves. Then, the survey technologies such as seismic surveys are used to discover the amount

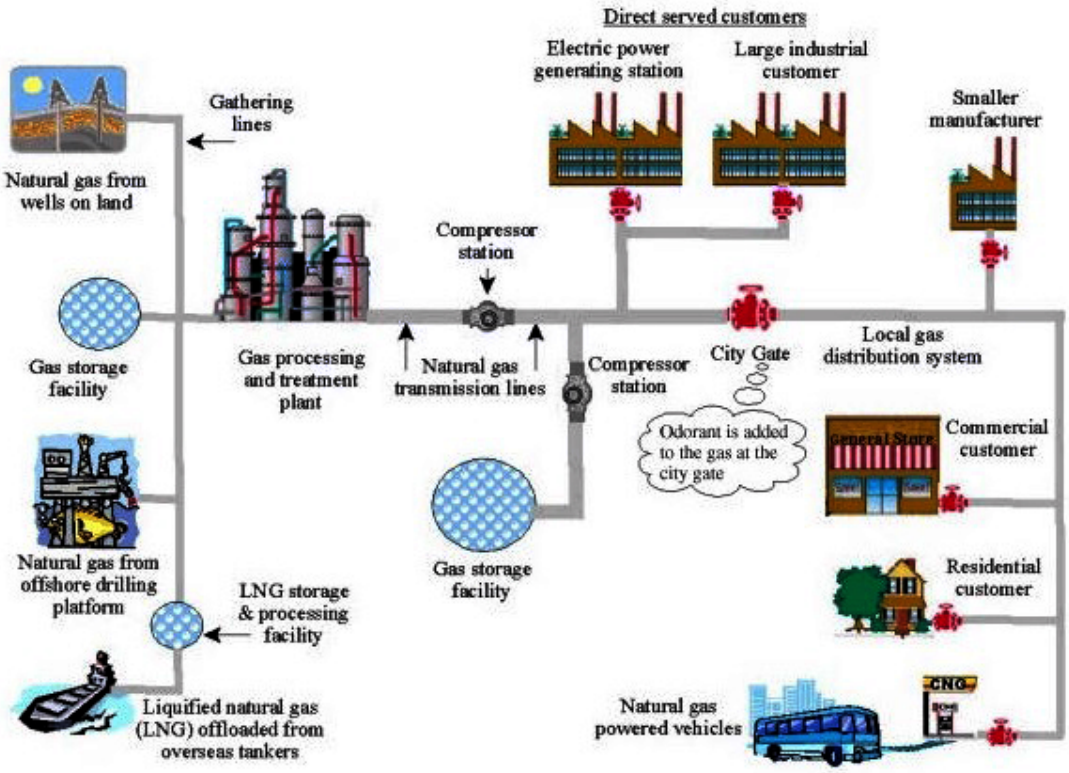

Figure 1.1: Natural Gas Supply Chain

of oil and gas deposits in the rocks. Also, the test wells are drilled in the area of interest to test the quantity and quality of the gas resource.

2. **Extraction:** To drill usually multiple wells to access the gas reserves. These wells are either vertically or horizontally drilled, depending on where the gas reserves lie within the coal seam, shale or tight sandstone. The Coal Seam Gas (CSG) reserves are easier to access, while shale and tight gas reserves are deeper underground and therefore hard to access.

3. **Production:** The natural gas is transported from the well to the refinery to be refined by the gathering line. This stage includes some processing facilities, such as removing water, carbon dioxide or sulfur.

4. **Transmission:** The natural gas is transported over long distances through the high-pressure pipeline from gathering, processing factory or storage facility to the distribution system.

5. **Storage:** This stage accounts for the storage of natural gas. It is one of the new and critical steps of the natural gas network process that must respond to the demands of different periods of the year and unexpected events, such as natural disasters.

6. **Distribution:** In this part, natural gas is transported by the low-pressure pipeline from gas pressure reduction station (City Gate Stations) to the end users.

7. **Marketing:** This is the final stage and involves the buying and selling activities in the market.

In the next section, some of the problems arising in the above stages are outlined in more detail.

1.3 Optimisation in Natural Gas Industry

The process of moving natural gas from the gas fields to consumers is complicated and requires the resolution of a number of problems from the extraction to the processing and distribution of products. The methods of Operations Research have been successfully applied to a number of problems arising in each of the four main stages of the gas industry namely: the natural gas production, transmission, distribution, and marketing. We discuss optimisation problems in the natural gas industry in more detail with the focus on these four stages. New designs or expansion in the capacity of the main components are required to increase natural gas consumption as one of the cleanest fuels.

1.3.1 Optimisation in Gas Production

There are many different optimisation models in the natural gas recovery area. According to Zheng et al. (2010), production scheduling, placement of the well-head, gas recovery systems, and facilities design are some of the related problems in the production stage of the natural gas industry. Also, Zheng et al. (2010) described the two following optimisation problems in this area.

Production Scheduling Considering Well Placement

The gas reservoir is accessed by drilling multiple wells on the surface, the pressure of the gas will reduce at all wells drilled on the same reservoir when the gas is withdrawn from any of the wells. So, the withdrawal rate for every drilled well will decrease at each period. The optimal production scheduling problem is to find the optimal well location and withdrawal rate at every drilled well at each period. For example, Murray III and Edgar (1978) investigated optimum production scheduling including the determining the location of wells either with the design of gas production scheduling or the design of gas wells and gathering systems.

Total Gas Recovery Optimisation

Using water flooding is one option for withdrawing the natural gas from the reservoir that leads to the following immediate problems: to find optimal water injection places with respect to different objectives, such as the maximal ultimate recovery, or the total revenues. Many optimal control models have been proposed for this problem (Zheng et al. (2010)).

1.3.2 Optimisation in Gas Transmission Networks

The problems in the optimisation of gas transmission networks can be divided into four groups: optimal design, optimal flow, optimal operation, and optimal expansion. We briefly review the models and methods of optimisation work in the area of gas transmission that is the starting point of our work. Before identifying the related problems, we need to consider the connected components, model characteristics, and network topologies in the gas transmission line. Figure 1.2 presents the physical components of a gas transmission network with a combination of parallel and series configurations for compressor stations. We need to note that we have permission from Geißler et al. (2013) and Springer as the publisher of this paper to use Figure 1.2.

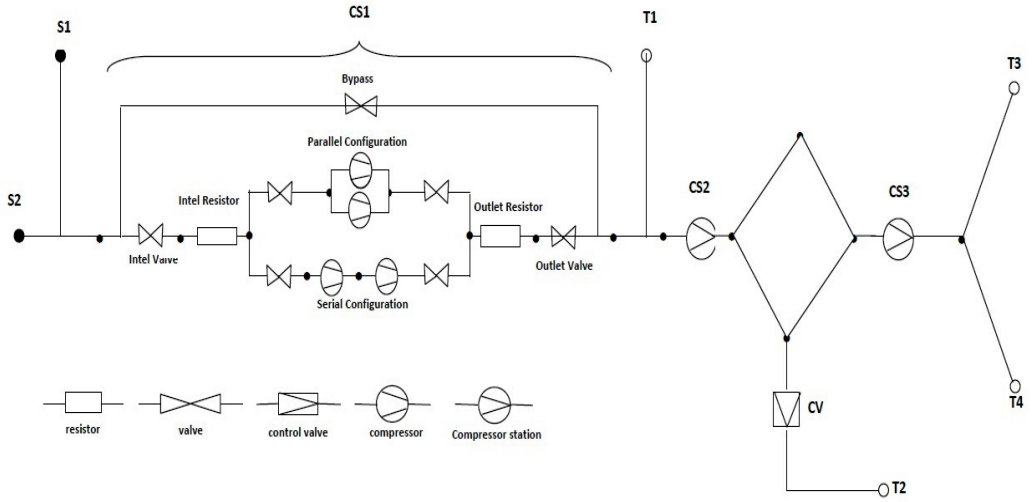

Figure 1.2: Gas network transmission with two configurations of compressor stations.

Components of Natural Gas Transmission Networks

- **Pipelines:** The pipeline in the gas transmission network are known as arcs. Some arcs include the compressors stations. Therefore, two sorts of arcs have been considered: passive arcs correspond to pipelines, and active arcs correspond to pipelines with compressors (De Wolf and Smeers (2000)). The mainline transmission pipelines are usually large size, between 16 and 48 inches in diameter.

- **Resistors:** The gas networks contain further elements such as controllable elements like measuring stations, filtration plants, gas-preheats and gas-coolers which cause a loss of pressure. These elements come under the term resistors. The resistors are located beside the pipe and are divided into the set of resistors with variables, such as flow dependent pressure drop, and the set of resistors causing a fixed loss of pressure (Geißler et al. (2013)).

- **Compressor stations:** Pressure is lost due to the friction between natural gas and the inner walls of the pipeline. Also, the natural gas volume is reduced because of different temperatures between the pipeline and environment. Compressor stations are located along transmission lines that consist of a parallel or series or a combination of both with several compressor units

to hold the pressure and increase the capacity of the gas transmission line (Hamedi et al. (2011)). There are three types compressor stations: active if the discharge pressure is greater than suction pressure, bypass, if the discharge pressure is equal to suction pressure and closed, if there is no flow rate (Cobos-Zaleta and Ríos-Mercado (2002)). The compressor station usually works at a pressure of approximately 200 psi to 1400 psi.

- **Valves:** Valves are for stopping the gas flow in situations such as maintenance or replacement for certain parts of the pipeline. Valves can either be open or closed. An open valve causes no pressure drop, while a closed valve restricts gas from passing.

- **Control Valves:** These are located at such transition points to reduce pressure manually when gas is transported from a large conveyor pipeline into a local sub-network. According to Geißler et al. (2013), the control valve station has three base stages:

  1. It might be active, which means the bypass valve is closed and the inlet and outlet valves and the control valves are open.
  2. It might be in bypass mode, which means the inlet and outlet valve are closed and the bypass valve is open.
  3. The station might be closed, which means all three valves are closed.

Model Characteristics

There are different assumptions related to each defined problem on a natural gas network. Some of these assumptions are detailed below. The resulting constraints affect the problem's complexity and formulation. Some of the most usual attributes of the natural gas networks problems can be illustrated as follows.

- **Steady State or Transient State**

Two main categories of the state of the natural gas pipeline are steady state and transient state. These show how the gas flow changes over the time. In the steady state system, the gas flow is determined with some values that are independent of the time constraints of the system, and nonlinear algebraic equations describe the pipeline gas flow. In contrast, in the transient state, the system

variables, such as mass flow balance through the pipelines and gas pressure levels at each node, are defined as the functions of the time (Hamedi et al. (2011)).

Transmission Network Topology

There are two different network types of topologies: (a) linear or gun-barrel, and (b) cycle. To identify the natural gas transmission networks topologies, Wu et al. (2000) explained the usual methodology as follows: In a given network, to begin with, remove the compressor arcs. Then, unite the remaining connected components into a big super-node. Finally, the compressor arcs are moved back to normal. The new network is called an associated reduced network. We have permission from Springer as the publisher of paper Ríos-Mercado et al. (2002) to use Figures 1.3 and 1.4.

- **Linear Topology:** This occurs when the compressors are arranged in a path, that is when the reduced network is a single path. Also, when the compressors are arranged in branches in the network, the reduced network is a tree. Figure 1.3 shows a network and the associated reduced network.

- **Cyclic Topology:** This occurs when the compressors are arranged in a cycle design with other compressor stations. In other words, it refers to a cyclic reduced network. Figure 1.4 shows a network and the associated reduced network.

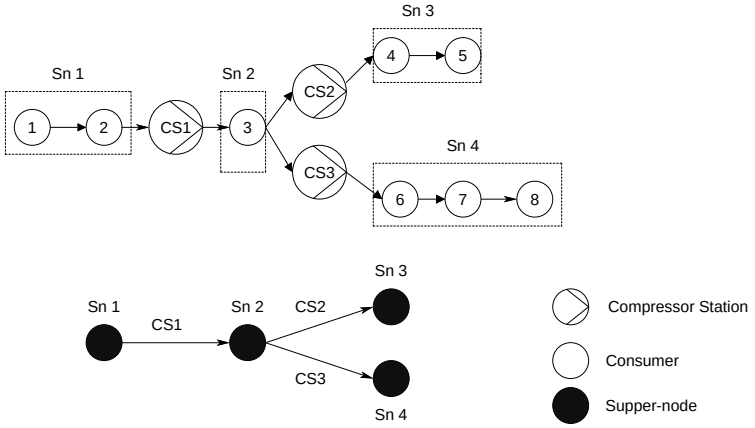

Figure 1.3: Linear Topology, a transmission network and the associate reduced graph

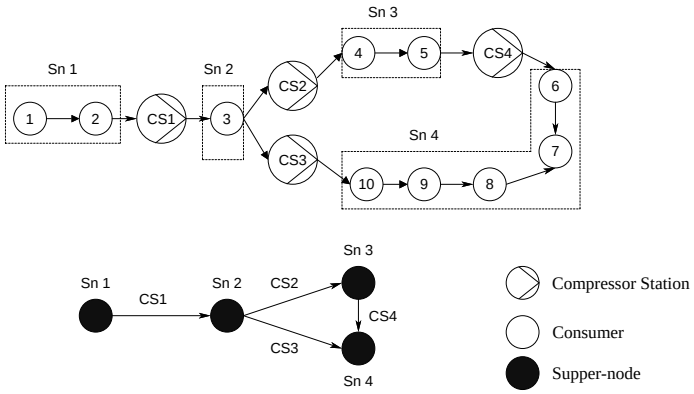

Figure 1.4: Cyclic Topology, a transmission network and the associate reduced graph

Transmission Network Design

The primary design variables of essential components of the natural gas transmission networks include pipelines and compressor stations over a planning horizon. The design characteristics of pipelines include pipeline diameter, pressure, and flow rate. Moreover, the design variables of compressor stations are location, suction pressure, pressure ratio, station throughput, fuel consumption and station power consumption (Hamedi et al. (2011)). The objective function in the network design is to minimise the investment cost.

Transmission Network Flow

Minimising the cost along with providing enough services to customers are the principal objectives for network flow optimisation. Also, the volume of gas flowing through the network elements is the decision variable for the network flow problems, such as minimum cost flow problems, shortest path problems, and maximum flow problems (Hamedi et al. (2011)).

Transmission Network Operation

Many factors can influence the operation costs for high pressure. These factors include operation costs (which consist of the fuel cost, cost of turning them on and off, and maintenance cost for each compressor), penalty cost (which is

acquired if the compressor does not run for a specific period and avoids frequent start/stop actions), and consumer demand (Uraikul et al. (2004)). Compressor stations use 3% to 5% of the amount of natural gas as fuel from their lines. Optimisation techniques can save up to 20% on the fuel costs that are used by compressor stations (Hamedi et al. (2011)). Thus, the main problems regarding the operation of gas transmission networks are fuel cost minimisation and the minimising the fuel consumption of compressor station.

Transmission Network Expansion

The optimal expansion is another type of design optimisation problem while it optimises an existing network. Timing, sizing and location decisions are some of the considerations for future expansion for optimal expansion capacity (Tabkhi et al. (2009)). The addition of compression capacity to a pipeline is the most common form and most straightforward method for increasing capacity. This capacity is increased by adding new compressor units at an existing site, to provide more compression and provide more reliability. Looping is another significant alternative to additional compression where installing a parallel pipeline close to the existing line to lower the flow resistance and then drop the pressure to increase the throughput and increase the available line pack. The pipeline looping is very flexible from a design point of view, as the length, the diameter and the location of each loop section can present a safety threat to the existing pipeline. Also, this has been used on systems where higher flow reliability is required than that which can be expected from compression capacity increases (Tabkhi et al. (2009)).

1.3.3 Optimisation in Gas Distribution Networks

Natural gas distribution lines are located next to a transmission line that transports natural gas over the pipelines from the gas pressure reduction stations to the end users. The optimisation problems in the gas distribution networks arise from the pipeline design and operation of the gas distribution systems. The operation problems can also arise in the combination of natural gas system and electricity system. Before we identify the related problems, we explain the connected components and network structures in the gas distribution network.

Components of Natural Gas Distribution Network

The main components of this system are:

- **Pipeline**: Natural gas distribution systems include small to mid-size pipelines (ranging from 2 to 20 inches in diameter) which are constructed out of copper, plastic and cast iron. They are usually installed underground along streets and roadways in the tree or acyclic structure designs. Due to security reasons, distribution pipelines typically operate below their capacity and work at a pressure of approximately 0.5 psi up to 200 psi (Ríos-Mercado and Borraz-Sánchez (2015)).

- **Pressure Reduction Stations**: The high gas pressure is delivered using transmission lines to the pressure reduction stations which reduce the pressure to that required at the demand points. The pressure of the gas is reduced three times as follows:

    1. **City Gate Stations (CGSs):** The CGS is located at the border of the transmission line and the distribution line for three purposes. First, it reduces gas pressure from 1000 psi to 250 psi. Secondly, a distinct sour odour associated with natural gas is added for the consumer to be able to smell the balance of quantities of gas. Finally, they measure the flow rate of the gas to determine the amount of receiving gas by public servers (Australian Energy (2009)). According to Ranjbar (2011), the different devices used in CGS are
        - Insulation joint,
        - Odorise,
        - Scrubber,
        - Filter separator,
        - Dry gas filter,
        - Isolating valves,
        - Indirect water bath heater,
        - Safety shut off valve,
        - Pressure reducing and control valve (throttle valves),
        - Metering system,
        - Safety valve,

- Sound attenuator,
- Sensing line,
- Drain valves,
- Supports,
- Electrical systems.

With gas pressure reduction, the temperature drops (Joule-Thomson effect). The dropped temperature may cause freezing of water vapour in natural gas, which results in swelling and corrosion in the pipeline. The solution is to preheat the natural gas before pressure reduction. While the gas pressure decreases in the expansion valves, a significant amount of energy will be wasted (Sanaye and Nasab (2012)). Combined heat and power (CHP) system can be used to gain even more electricity and savings. The CHP system generates electricity and heat simultaneously using a single source of fuel (Sanaye and Nasab (2012)). Therefore, in this way, a significant part of the consumed energy in the compression station is recovered.

2. **Town Board Station (TBSs):** TBSs provide the second pressure reduction station and reduce the pressure from 250 to 60 psi. The devices used in the TBS are the same as those in CGS except filtering, odorising, and heating. District Regulating Stations (DRSs) are a kind of TBS which reduces pressure for not only commercial consumers but also for high demand consumers such as universities and hospitals.
3. **Regulating Stations:** They adjust gas pressure in the network to ensure gas is delivered at a suitable pressure for users. Regulator service will reduce the pressure to less than 1/4 psi, which will be set up in each consumer property.

- **Consumers:** There are three different types of consumers which have been outlined in more detail in Section 1.1.3.

Consumer type A: Electricity generators,
Consumer type B: Industrial,
Consumer type C: Residential and commercial.

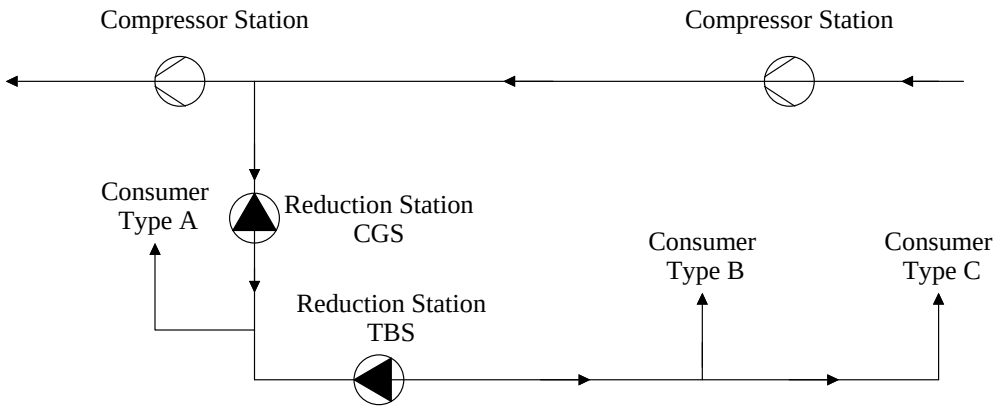

Figure 1.5: Fragment of gas distribution networks with different groups of consumers.

Figure 1.5 shows the process of gas distribution network from transmission network to the consumers. We need to note that we have permission from Szoplik (2015) to use this figure in our thesis. Medium gas pressure is transported by a pipeline from CGS to the consumer type A (generate electricity power) or TBS. In the second stage, the low gas pressure moves from TBS to the regulator station. Finally, the regulator reduces the pressure of gas according to demand satisfaction, and moves the gas to the consumers as the last node in the natural gas distribution networks. We need to note that we have permission from Springer as the publisher of paper Szoplik (2015) to use this figure.

Gas Distribution Network Structures

There are two basic structures for gas distribution networks normally the tree and the loop network. We detail each below. Moreover, the advantages and disadvantages of these two structures are presented below.

- **Tree Structure**: It can be defined as a series of pipes in which each node is connected to the source by only one route. In this structure, each consumer node is served by only one node. This serving node could be a source node or another consumer.

  The advantages: It is straightforward to find the flow through each pipeline over a tree network through a system of linear equations. Also, the

other variables such as pressure at each node and pipeline diameter of each link can be obtained from the physics gas equation. The tree structure is mostly a cheaper choice of design of the network because all nodes are connected with the minimum number of links.

The disadvantages: The failure of a single link will prevent a subset of consumers from being served. In the tree structure, each consumer is connected to the source with only one path.

- **Cycle (Loop) Structure**: It is an acyclic structure. They are designed as a series of interconnected closed loops. We note that usually, the acyclic structure is called a loop structure in the literature. This structure guarantees the delivery of gas to the users under single pipe failure conditions.

  The advantages: There are at least two routes to serve consumers. In fully looped systems, each demand node can be supplied from the source(s) through at least two independent paths. Two supply paths are said to be independent if they do not have a pipe in common. However, they can have a node in common.

  The disadvantages: Mostly, the loop structure design is more expensive than the tree one, and the network analysis is complicated (flow and pressure).

We can divide the problems arising from the gas distribution network into three parts as design, operation, and expansion. The main objective of the problems arising from different parts of the gas distribution is to meet customer requirements at minimum total cost.

Distribution Network Design

Pipeline systems in the natural gas network must be designed based on gas flow rate, length of pipe, maximum gas pressure drop allowance and maximum gas velocity. We identify more specific problems as follows:

- **Fundamental Problem (Problem 1)**

  Network Design and Allocation Problem: Given a set of nodes and customer requirements, determine the optimal network. This involves determining the

- Network layout,
- Diameter of the connected links,
- Pressure at each node,
- Flow through each link.

The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

There are only a few papers on the fundamental problem, but most of the literature on gas distribution networks has focused on the following subproblem.

- **Subproblem (Problem 2)**

  Allocation Problem: Given a set of nodes, network structure and customers requirements, determine the optimal pipeline design of the network. This involves determining the
  - Pipeline diameter for each link,
  - Pressure at each node,
  - Flow through each link.

  The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

According to Shiono and Suzuki (2016), there are three cases with respect to the diameter of each link,

- The Single Diameter (SD), where each link consists of one commercially available diameter.
- The Multi-Diameter (MD), where a link consists of different commercially available diameter segments that are serially connected.
- The Continuous Diameter (CD), where a link consists of one diameter type that should be determined between the given minimum and maximum values.

Distribution Network Expansion

The investment in the expansion of gas distribution networks involves capital works to improve and expand the capacity of the existing networks into new residential and commercial developments, regional centres and towns. Also, the cost of distribution investment depends on a range of factors consisting of the distance of new infrastructures from entry points on the gas transmission line or distribution line, the density of housing, and the attendance of other industrial and commercial customers in the area (Australian Energy (2009)). The following objectives for the expansion of gas distribution networks can be outlined (Shiono and Suzuki (2016)):

- Minimise the sum of reinforcement costs.
- Maximise the expected net present value (NPV) which comprises of generated revenue and various costs such as the cost of removing old infrastructure, operating cost to maintain and operate infrastructure and the production costs on different fields.

Optimisation in the Combined Natural Gas System and Electricity System

The electricity generators are one of the primary uses of gas. Combined-cycle plants are a type of power plant with high efficiency with less damage to the environment. It is directly linked to the medium pressure line. Hence, the electricity and gas networks are connected. There are some related optimisation applications regarding this relationship. For example, Rubio-Barros et al. (2008) provided a complete survey of state of the art in the combined operational planning of natural gas and electric power systems.

Saldarriaga et al. (2013) presented a new distribution expansion model that considers electricity distribution networks and natural gas distribution networks as one system with a high penetration of distributed generation based on natural gas. The objective function is the present value of the following six terms:

1. The cost of installing new elements of the electricity networks.
2. The cost of expanding the capacity of existing elements of the electricity networks.

3. The investment cost of the new CGS station of the distribution networks.
4. The cost of expanding the capacity of existing city gates of gas distribution networks.
5. The cost of installing and increasing the capacity of the natural gas distributed generation.
6. The operative cost of the electricity network.

1.3.4 Natural Gas Market Model

The optimisation of natural gas market models has been divided into two groups: regulated and deregulated gas markets (Zheng et al. (2010)). The aim is to consider how alternative environmental and energy policies, as well as infrastructure investments, could affect public objectives such as consumer costs, overall market efficiency, energy security, and environmental impacts Ruiz et al. (2014). We detail more below:

Regulated Market

A variety of participants between the original producer and end user exists, such as the gas producers, the gas pipeline companies, local gas distribution companies and consumers. In a regulated market, gas prices in each transaction between these participants are tightly regulated by government policy (Zheng et al. (2010)). O'Neill et al. (1979) proposed a model for allocating gas to the users with different priorities under the government regulations when a gas shortage emergency arises. In their model, all users are divided into nine categories with priorities 1 through 9 (from low to high order).

Deregulated Market

The deregulation of the gas market not only changes the roles of the former participants but also helps to create more participants, such as gas marketing companies. An example of a series deregulation policies is to change the traditional role of pipeline companies as owners of natural gas in which the buyers can transport their gas through the pipeline system by paying some fees (Zheng et al. (2010)). Gabriel et al. (2005) aimed to minimise the cost or maximise the profit of each participant. They considered six types of participants: the pipeline op-

erators, the production operators, the marketers/shippers, the storage reservoir operators, the peak operators and the consumers.

In section 1.4 we give a brief review and outline of the thesis.

1.4 Review and Outline of Thesis

In this thesis, we consider the network design and allocation problem for gas distribution networks. The fundamental problem is:

- Given a set of nodes and customer requirements, determine the optimal network. This involves determining the network layout, the diameter of the connected links, the pressure at each node, and the flow through each link.

The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which is tree. There are three cases of pipeline diameters to consider: single diameter; continuous diameter; and multi diameter. In the literature, the cases of single diameter and continuous diameter have been considered by Rothfarb et al. (1970) and André et al. (2013). There is no work in the literature that considers the case of multi diameter. The contributions of our research are in the area of gas distribution network design and allocation that have unresolved issues. More specifically we develop an accurate Mixed Integer Non Linear Programming (MINLP) model for the fundamental problem in the case of multi diameter. The multi diameter case corresponds to where a link consists of different commercially available diameters that are serially connected. Our MINLP model is computationally difficult to solve, particularly for larger networks. Thus, we are motivated to develop two different solution methods for our model. One is an approximation method and the other a heuristic method.

This thesis is divided into six chapters that are organised as follows.

Chapter 1 starts with an overview of natural gas including composition, formation, and uses of natural gas. Also, the gas supply chain that is the process of moving natural gas from the gas field to consumers is presented. The problems

arising in the four stages: production; transmission; distribution; and marketing of gas supply chain are discussed. In particular, we explain gas distribution line including components, structures, problems in design and expansion of gas distribution networks.

Chapter 2 begins by providing the background of the problems for the design of gas distribution networks. We consider two main problems in the area of gas distribution network design: the fundamental problem which is network design and allocation problem and the sub-problem which is the allocation problem. We provide the mathematical models as well as the solution approaches related to these two problems from the literature. We present the literature review for water distribution networks as well. The design of the gas and the water distribution networks are similar in terms of their mathematical modelling. Most of the literature on distribution networks has focused on the problem where the network structure is given (the allocation sub-problem). In contrast, there are only a few papers on the problem that involves the determination of the network structure and its components (the fundamental problem). Our motivation in this thesis is to address this problem.

In Chapter 3, we develop an effective MINLP model to design the gas distribution network (tree structure) and to allocate the multi diameter types for each link (Fundamental Problem). The objective function is a linear cost function of the pipe diameters. We consider a wide range of design parameters including the number of supply and demand nodes, the set of pipe diameter types, the length (distance) between nodes, and the pressure limits at each node. We consider the constraints under steady-state conditions such as pressure limits at each node, the flow balance through each link, the pressure drop equation for two ends of each link. The decision variables are the selected links connecting nodes, the flow through each link, the pressure at each node, and the length of selected diameter for each link. Our model includes a linear objective function and the nonlinear and nonconvex equality constraints. These constraints make our MINLP model computationally difficult. We develop two different solution methods for our model. One is an approximation method and the other a heuristic method.

To solve our MINLP model using the approximation method, we need a

method that handles the nonconvexities of the constraints. The algorithm that we use is Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method given by Viswanathan and Grossmann (1990). The advantage of the OA/ER/AP algorithm, among the other MINLP algorithms, is in its ability to handle the nonconvexity of the problem. Our contribution in this part is that we review this algorithm and then examine its functionality and efficiency on our MINLP model. The following details are given in Chapter 3.

- Our new MINLP model including the notation and terminology.
- The Outer Approximation algorithm and its variations.
- The OA/ER/AP method.
- The effective parameters and stopping criteria for the OA/ER/AP algorithm that influence the feasibility and efficiency of the solution for our model.

In Chapter 4, we test our model using the OA/ER/AP algorithm presented in Chapter 3. We consider the effective choice of parameters and stopping criteria for the OA/ER/AP algorithm that facilitates the finding of a cost-effective, feasible solution in a reasonable amount of computation time (as our model is computationally difficult to solve using the OA/ER/AP algorithm, we allow a maximum 24 hours). The OA/ER/AP algorithm alternates between two subproblems including a NLP sub-problem and the Master MILP sub-problem. In the NLP sub-problems, the integer variable of our model are fixed. In the master MILP sub-problem, the effect of nonconvexities of our model will be reduced by linearization of the nonlinear functions of our model at the set of linearization points. The number of starting points in the NLP sub-problem and the value of the optimality gap in the Master MILP sub-problem are the main inputs that influence the feasibility and efficiency of the solution for our model. A single start implementation might be trapped in a local optimum, but the multiple starts provide a better chance of finding a cost-effective feasible solution. We generate different sized networks (5, 10, 15, 25 and 50 nodes) to test our model (30 instances for each network size). In this chapter, the OA/ER/AP algorithm is applied to our test cases and we give the following details:

- The computational results with the Single start for the NLP sub-problems.

For the medium and large sized problem, we also consider different values of Relative Optimality Tolerance (ROT) for the Master MILP sub-problem.

- The computational results with the Multi start for NLP sub-problems. For the medium and large sized problem, we also consider different values of (ROT) for the Master MLIP sub-problem.
- Comparative analysis between the NLP-Single start and the NLP-Multi start strategies.

From the computational results using the OA/ER/AP algorithm, it turns out that the smaller sized networks are solved when the optimality gap is 10^{-13}. We consider a greater value than 10^{-13} for the optimality gap (the value of ROT(MILP)) for networks with 25 and 50 nodes. We test all 30 instances for specific values of ROT(MILP) under the NLP-Single start and the NLP-Multi start conditions to find a cost-effective feasible solution within 24 hours. As expected the NLP-Multi start yields an improved objective function value but requires more computational time than the NLP-Single start. Our main outcome is a MINLP model that is effective for small networks.

In Chapter 5, we develop a new heuristic algorithm for solving the MNLP. Our heuristic algorithm generates a good feasible solution for the gas distribution network design and allocation problem (the Fundamental Problem). We aim to determine the gas distribution network structure and its components. These components are the different pipe diameter types (multi diameter) for each link, the suitable pressure at each node, and the flow through each link under steady-state conditions. Our heuristic algorithm solves the fundamental problem in a reasonable amount of computation time. Our solution strategy is to reduce the level of difficulty by converting the MINLP problem to a Linear Programming problem generating the integer variables in the outer level separately. Thus, our proposed algorithm includes two levels: in the outer level, a tree network is generated, and in the inner level, we determine different diameters for the given links as well as determining the pressure at each node and the flow through each selected link. This method is implemented in JAVA as a programming language, AIMMS as a commercial software package, and Excel as a computer package with arithmetic operations and functions. The quantitative analysis of the effects of different parameters and stopping criteria of our heuristic algorithm on optimal decisions is also investigated. The effective parameters included the

values of the sample size, elite sample size and smoothing parameter and the stopping criteria included the value of tolerance parameter and integer parameter. This chapter includes computational results for our algorithm. We also present a comparative analysis of the objective function value and computation time between our heuristic algorithm and the AOA algorithm. Our main outcome is a heuristic algorithm that finds a good quality feasible solution for the large sized test problems (50 nodes) within 4 hours. The results obtained from the comparative analysis of the objective function value are:

- Our heuristic algorithm improves the average objective function value for small sized test problems, by 2% and 3.57% on the NLP-Single start and the NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for medium-sized test problems, by 33% and 28% on the NLP-Single start and the NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for large sized test problems, by 32% and 27% on the NLP-Single start and the NLP-Multi start, respectively.

We also note that an average computation times (seconds) of our heuristic algorithm for networks with 10, 25, and 50 nodes of 765.33, 4,351.46, and 10,162.53, respectively. The following results are obtained from the comparative analysis in the computation time for networks with 25 and 50 nodes:

- Our heuristic algorithm improves the average computation time for medium sized test problems, by 135% and 143% on the NLP-Single start and the NLP-Multi start, respectively (when the optimality gap is 0.01).
- Our heuristic algorithm improves the average computation time for large sized test problems, by 232% and 188% on the NLP-Single start and the NLP-Multi start, respectively (when the optimality gap is 0.1).

Finally, Chapter 6 highlights the major contributions of the study. We provide a summary of the earlier chapters and conclusions obtained from the research. We conclude the chapter with some suggestions for possible directions for future research in the area.

Chapter 2

Models

In this chapter, the background of gas distribution network design is provided. We first establish an overview of problems for both gas and water distribution networks. Then, we detail the trend of the research in gas distribution network design under steady state conditions. The mathematical models and solution methods will be discussed. Moreover, for more clarification of the problems in network design, we include the literature review for water distribution networks as well. The design of gas and water distribution networks are similar in terms of their mathematical modelling. Most of the literature on distribution networks has focused on problems when the network structure is given (normally, as a tree or a cycle (loop) network). In contrast, there are only a few papers on problems that also involve the determination of the network structure. The contributions of this thesis are identified based on the gaps in the literature. Our motivation in this thesis is considering the network design and allocation problem for gas distribution networks. Our main problem is:

- Given a set of nodes and customer requirements, determine the optimal network (minimum cost). This involves determining the network layout, the diameter of the connected pipeline, the pressure at each node, and flow through each link.

The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, the pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

This chapter is organised as follows: Section 2.1 presents the background of the gas distribution network design including the main problems of gas network design. In Sections 2.2, we present the literature on the design of gas and water distribution networks when the structure is given as a tree or a cycle (loop) network. In Section 2.3, we review the literature on the design of gas and water distribution networks when the structure is not given. In Section 2.4, the research gap and the research problem are provided. The contributions of this thesis are provided in Section 2.5.

2.1 Background

In the last few decades, many problems in the gas industry have been tackled by operations research models and techniques. As we have already discussed in Chapter 1, these problems cover the area of production, transportation, distribution, and marketing. Firstly, we direct the reader to several excellent survey papers of optimisation techniques in the gas industry. Then, in particular, we present some background of optimisation techniques in gas distribution networks.

Nikbakht et al. (2012) completed a literature review on natural gas supply chain modelling and designed a multi-echelon supply chain for the natural gas transmission system. Zheng et al. (2010) focused on three specific aspects of the gas industry: production, transportation, and marketing. They considered the mathematical modelling of six general problems including the production scheduling problem, the maximal recovery problem, the network design problem, the fuel cost minimisation, and the regulated and deregulated market problems. Also, they provided a literature review of existing optimisation techniques as the methods of finding the solutions to these six problems.

Hamedi et al. (2011) introduced some problems related to the design, flow, operation and expansion of gas transmission networks. Ríos-Mercado and Borraz-Sánchez (2015) presented a state of the art survey focusing on specific categories that include short-term basis storage (line-packing problems), gas quality satisfaction (pooling problems), and compressor station modelling (fuel cost minimisation problems).

Recently, a survey of optimisation techniques in operation and design of natural gas pipelines has been provided by Demissie and Zhu (2015). This study covers the literature on gas transmission and distribution networks from 2000 to 2014 (Demissie and Zhu (2015)). They also concluded that gas distribution systems has received less attention from researchers than transmission systems.

In terms of optimisation techniques in gas distribution networks, there has been very little work in the literature (Demissie and Zhu (2015)). For more clarification of the problems and applications of optimisation techniques in the field of network design, we have gone through the literature of the design of the water distribution networks as well. As noted earlier there is a strong similarity in the mathematical modelling of gas and water networks. Both have flows and both have complex constraints. Gas networks have pressure requirements and water networks have water height as a factor which plays a similar role (De Wolf and Smeers (1996)).

Generally, in the literature, the network design for the gas or water has been presented in terms of graph theory. The vertices (nodes) of the graph correspond to the points such as stations and consumers to be linked. The arcs represent the potential connection between nodes such as pipelines. In most applications, the required network has a tree or cycle structure that is the underlying graph is a tree or a cycle (loop). The structure is motivated by the cost consideration. In general, network design is a very difficult problem due to the need to design reliable, resilient, and survivable networks when ensuring many design constraints (André et al. (2013)).

The main problems in the gas or water networks are designing optimal structures. In particular, this requires determining the diameter of each pipe in the network. We summarise these problems in more detail below:

- **Fundamental Problem (Problem 1)**

  Network Design and Allocation Problem: Given a set of nodes and customer requirements, determine the optimal network. This involves determining the

  - Network layout,

- Diameter of the connected links,
- Pressure at each node,
- Flow through each link.

The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

There are only a few papers on this fundamental problem. Most of the literature on gas distribution networks has focused on the following sub-problem.

- **Sub-problem (Problem 2)**

  Allocation Problem: Given a set of nodes, network structure and customers requirements, determine the optimal pipeline design of the network. This involves determining the
  - Pipeline diameter for each link,
  - Pressure at each node,
  - Flow through each link.

  The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

There are three cases to consider with respect to the diameter of each link (Shiono and Suzuki (2016)):

- The Single Diameter (SD), where a link consists of one commercially available diameter.
- The Multi Diameter (MD), where a link consists of different commercially available diameter segments that are serially connected.
- The Continuous Diameter (CD), where a link consists of one diameter type that should be determined between the given minimum and maximum values.

In this section we provide models and solution methods for the above problems. Briefly, we classify Problems 1, 2 as shown in Figure 2.1. We review the

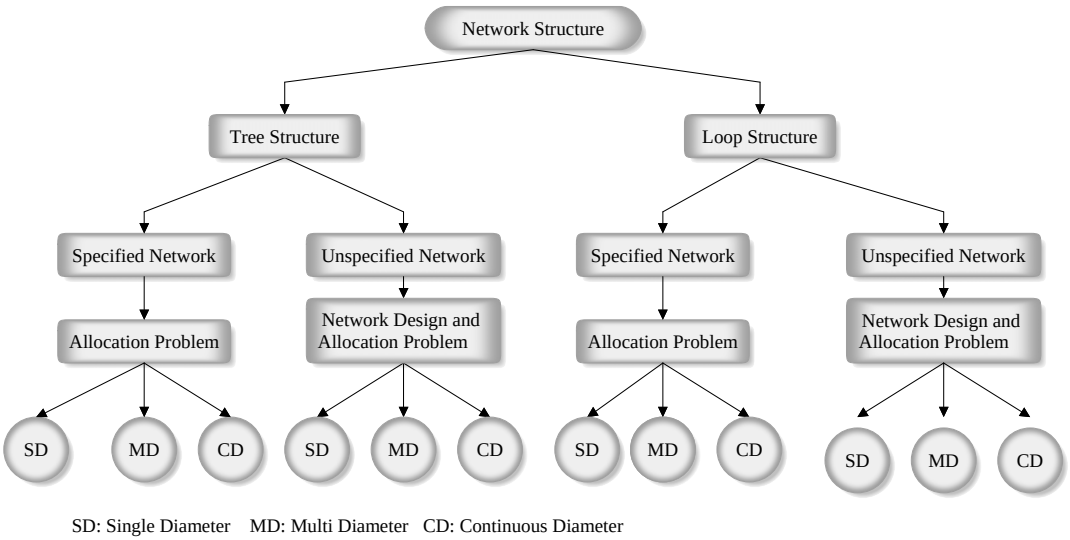

Figure 2.1: The classification for reviewing the literature according to the problems of network design and allocation problem

literature based on the classification given in Figure 2.1.

We distinguish the specified and unspecified network cases in our mathematical modelling using following notations:

- When the network is specified, we define the $n \times n$ adjacency matrix $A = [a_{ij}]_{n \times n}$ as an input parameter,

$$a_{ij} = \begin{cases} 1, & \text{if } i \text{ is a serving node for } j \text{ (i} \rightarrow \text{j)}. \\ 0, & \text{otherwise.} \end{cases}$$

- When the network is not specified, we define z_{ij} as a decision variable,

$$z_{ij} = \begin{cases} 1, & \text{if } i \text{ is a serving node for } j \text{ (i} \rightarrow \text{j)}. \\ 0, & \text{otherwise.} \end{cases}$$

We use the following notation and terminology to present the mathematical models in the literature.

Sets:

$N = \{1\} \cup N_c$: The set of supply and demand nodes.
$N = \{1, 2, 3, \ldots, i, \ldots, n-1, n\}$

N_c: The set of demand nodes $\{2, 3, \ldots, i, \ldots, n-1, n\}$.

D : The set of pipe diameter type, $\{d_1, d_2, \ldots, d_{|D|}\}$.

O : The set of loops in the network.

Input data:

s_1 : The amount of supply at source node 1.

s_i : The amount of demand at node i.

$C(d)$: The cost of per unit length of pipe diameter type $d \in D$.

L_{ij} : The length between node i and node j.

π_i^L, π_i^U: Lower and Upper square pressure limits at node i.

D_{min}, D_{max} : The minimum and maximum value of diameter
when we consider continuous diameter of each link.

Variables:

pr_i : The gas pressure at node i.

π_i : The square of gas pressure at node i.

f_{ij} : The flow rate through the pipeline from node i to node j.

d_{ij} : The pipe diameter type for the link from node i to node j,

l_{ij}^d: The pipe length with diameter d from node i to node j.

δ_{ij}^d: The proportion of the length from node i to node j
that has a diameter type $d \in D$.

We define $d_{ij} \in D$, when we consider single diameter for each link (the first case) and $d_{ij} \in [D_{min}, D_{max}]$, when we consider the continuous diameter for each link (the third case). We note that we consider designing of the gas networks under steady state conditions. Our motivation is the fact that the steady state conditions can be applied widely in gas transportation, in particular for low-pressure gas distribution lines (Osiadacs and Pienkosz (1988)). A network is in the steady state conditions when the flow of gas in the system is independent of time. The problem of gas networks in steady state conditions is usually that of computing the values of node pressures and the values of flows in the individual pipelines for known values of lower and upper pressure limits and of gas supply and consumption at the nodes (Osiadacs and Pienkosz (1988)). Similarly, the steady state conditions can also be defined for water networks (Djebedjian et al. (2005)). We define the known objective functions of the allocation problem in the literature for cases of single diameter, multi diameter, and continuous diameter.

The Objective Functions

The objective function is to minimise the design cost of the network. The main cost of design is related to the cost of the pipeline in which the cost is increased when the pipeline diameter is increased. The known objective functions in the literature for the different scenarios are as follows:

- The single diameter case corresponds to where a link consists of one commercially available diameter ($d_{ij} \in D$). In this case, the objective function (Function (2.1)) is a discrete cost function of the pipe diameters.

$$\sum_{(i,j)\in A} C(d_{ij})L_{ij} \tag{2.1}$$

- The multi diameter case corresponds to where a link consists of different commercially available diameters ($d \in D$) that are serially connected. In this case, the objective function (Function (2.2)) is a linear cost function of the pipe diameters.

$$\sum_{(i,j)\in A} \sum_{d\in D} C(d)l_{ij}^{d} \tag{2.2}$$

- The continuous diameter case corresponds to where a pipe diameter is con-

tinuous ($D_{min} < d_{ij} < D_{max}$). Parker (2004) used regression analysis to estimate the relation between the total cost and pipeline diameter. He found that the per unit cost of the pipeline is calculated as a quadratic cost equation containing three terms of a_1, a_2, and a_3 (André (2010)). The objective function for this case is:

$$\sum_{(i,j)\in A} (a_0 + a_1 d_{ij} + a_2 d_{ij}^2) L_{ij}. \tag{2.3}$$

In particular, De Wolf and Smeers (1996) considered the values of the coefficients of the quadratic function (2.3) as $a_0 = 7.7476, a_1 = 7.4782 \times 10^{-3}$, and $a_2 = 2.5180 \times 10^{-5}$. They estimated these values based on the quadratic regression curve obtained from the real data (The relation between the total cost and pipeline diameter).

The Constraints

We first present the main known constraints that should be considered in any mathematical model for the design of gas and water networks. Then, we present the mathematical models for each case of diameter types in Table 2.1.

The following constraints are considered for the mathematical model of the design of the gas and water networks when the network structure is given under steady state conditions:

(1) Mass flow balance equation at each node.

$$\sum_{j\in A^+} f_{ij} - \sum_{j'\in A^-} f_{j'i} = s_i, \qquad \forall i \in N \tag{2.4}$$

where, $A^+ = \{j|(i,j) \in A\}$, $A^- = \{j'|(j',i) \in A\}$

(2) Pressure limits constraints at each node.

$$\pi_i{}^L < \pi_i < \pi_i{}^U, \qquad \forall i \in N \tag{2.5}$$

(3) Pressure drop equation through each pipeline.

$$\pi_i - \pi_j = \beta L_{ij} d_{ij}^{-5} f_{ij}^2, \qquad \forall (i,j) \in A \tag{2.6}$$

According to André et al. (2013), when we consider a fluid is flowing in a pipeline, the difference of pressure between two ends of pipeline finds its origin in the friction of the fluid on the internal wall of the pipeline. These energy losses, called head losses, depend on physical properties of the fluid (density and velocity) and the geometry of the pipeline (diameter, length and roughness). In the literature, there are several formulae of the head losses that differ according to the required degree of accuracy. The Weymouth equation (equation (2.6)) is the head losses formula that has been used in the literature for gas distribution system (Wu et al. (2007)). The Weymouth equation is referred to the pressure drop equation or the physics gas law in the literature. The Weymouth equation is a nonlinear and nonconvex equation that explains the relationship between flow, pressure and diameter of each pipeline (André et al. (2013)). The coefficient β in Equation (2.6) is a constant that can be computed by the following equations (Wu et al. (2007)):

$$\beta = K \times \lambda,$$

$$K = (1.3305 \times 10^5) \times CF \times SG \times T$$

where,

CF : The dimensionless compressibility factor,

T : The gas temperature,

SG : The gas specific gravity, and

λ : The frictional factor.

The equation (2.6) explains the capacity of each pipeline when f_{ij}, π_i, and d_{ij} are variables. Therefore, for the design of the gas network, we have to include the nonlinear equation as a constraint that explains the capacity of each pipeline. Also, the same equation can be defined for the water pipeline. The square of gas pressure is replaced by the water height at each node, the gas flow by the water flow and the value of coefficient β by different constant value. In the next section, we present the literature on the gas or water distribution networks design when the network is a tree structure.

2.2 Tree Structure

The tree and also the cycle (loop) networks are related to the different levels of reliability (level 1, level 2) for the demand nodes in the network. The reliability is the number of independent paths from the source node to the demand node (Afshar (2007)). The reliability of Level 1 means there is one independent path from the source node to each demand node. The reliability of Level 2 means there are two independent paths from the source node to each demand node.

In this section, we first present the literature on the Allocation Problem (Problem 2) in the given (specified) tree network. Then, we present the literature on the Network Design and Allocation Problem (Problem 1) when the structure of the network is not given (unspecified).

2.2.1 Specified Network

Much of the research in gas and water network design has centred on the determining pipeline diameter (Allocation problem) for specified tree network. In this case, as the network layout is defined, then the matrix of arcs is given as input data in the form of an adjacency matrix $A = [a_{ij}]_{n\times n}$.

The earliest known examples of the allocation problem have been mostly considered for given tree networks (the gas or the water). We note that whenever we have a tree network and if the direction of flow is given, the flow through the element of the network can be easily determined by solving the system of equations (2.4). Using the flow obtained from the system of equations (2.4), we determine the variables π_i and d_{ij} in the equation (2.6). Table 2.1 presents the mathematical models of the allocation problem (Problem 2) when we consider three cases for the diameter of the pipeline. The models 1 and 3 that are presented in Table 2.1 are Non Linear Programming (NLP) problems due to the nonlinear equation (2.6). Model 2 is a linear model when $L_{ij} = \sum_{d\in D} l_{ij}^d$. We need to note that the variables are π_i and l_{ij}^d. By replacing $L_{ij}d_{ij}^5$ with $\sum_{d\in D} l_{ij}^d d^{-5}$, the nonlinear equation (2.6) converts to a linear equation (2.7).

$$\pi_i - \pi_j = \beta(\sum_{d\in D} l_{ij}^d d^{-5}) f_{ij}^2, \quad \forall (i,j) \in A. \tag{2.7}$$

We also need to add the constraints $\sum_{d\in D} l_{ij}^d = L_{ij}, \quad \forall (i,j) \in A$ and $l_{ij}^d \geq 0$ in Model 2 that shows the summation of the length of pipelines with different diameters must be equal to the distance between nodes (i,j). We need to note that if $l_{ij}^d = 0$, then there is no diameter type d for link (i,j).

Single Diameter (SD)

In this case, the diameters of the pipe are collected from a finite discrete set of diameters for the tree network. Therefore, the mathematical model includes the integer variables that makes the problem computationally difficult. The earliest work for the case of single pipeline diameter was done by Bhaskaran and Salzborn (1979). They presented an Integer Programming (IP) mathematical model to determine the pipeline diameter of each link from the wells to the factory.

Bhaskaran and Salzborn (1979) also assumed that the pressure at the wells and the plants are given and the only variables are the discrete variable of diameters of each link (the MINLP Model 1 in Table 2.1 converts into an IP problem). To deal with the difficulty of integer variables in the model, they defined a new variable δ_{ij}^d as a portion of the length of each link that has a diameter type from the set of diameters. The variable δ_{ij}^d is aimed to consider the cases of single diameter. The objective function (2.1) turns into the following equation.

$$\sum_{(i,j)\in A} \sum_{d\in D} C(d) \delta_{ij}^d L_{ij}$$

Moreover, they added the new constraints for each link (i,j) to the model considering the following case:

$$\begin{cases} \sum_{d\in D} \delta_{ij}^d = 1, & \text{the sum of fractions is 1} \\ \delta_{ij}^d = \{0,1\}, & \text{for the case of single diameter.} \end{cases}$$

Table 2.1: Mathematical models for the Allocation Problem in the given tree gas distribution networks

		Objective Function	Constraints
Model 1	Single Diameter (SD)	(2.1)	$(2.4) - (2.6)$ $d_{ij} \in D, \qquad \forall (i,j) \in A.$
Model 2	Multi Diameter(MD)	(2.2)	$(2.4) - (2.5), (2.7)$, and $\sum_{d \in D} l_{ij}^{d} = L_{ij}, \qquad \forall (i,j) \in A.$
Model 3	Continuous Diameter(CD)	(2.3)	$(2.4) - (2.6)$, $d_{ij} \in [D_{min}, D_{max}], \quad \forall (i,j) \in A.$

When we consider constraint $\delta_{ij}^{d} = \{0, 1\}, \forall (i, j) \in A$, the IP model converts into a $0 - 1$ model. The model is implemented in APEX and tested with data from the Moomba gas field in South Cooper Basin in Australia.

In a recent paper, Shiono and Suzuki (2016) aimed to minimise the cost of a tree network determining the single diameter for each arc (i,j) (Allocation Problem). They considered the specified tree network with one single source. They eliminated π_i, π_j as variables in their model. The idea of elimination of the pressure as a variable is to consider the pressure drop through each path of the tree shape network that starts from the source node (node 1). The reason is that the supplied pressure π^U at the source node gradually decreases along the path, which means that the pressure at the last node in each path is greater than or equal to π^L. As a result, they replaced the equation (2.6) by the following equation that presents the pressure drop through each leaf starting from the source node that does not exceed $\pi^U - \pi^L$.

$$\sum_{(i,j) \in Path(1-v)} \beta L_{ij} d_{ij}^{-5} f_{ij}^{2} \leq \pi \quad \forall v \in V, d_{ij} \in D \tag{2.8}$$

where $V \subset N$ is the set of leaves in the tree network, $Path(1-v)$ is the $1-v-$path from source node 1 to $v \in V$ and $\pi = \pi^U - \pi^L$. The variables f_{ij} through the element of the network can be easily obtained from the tree structure using equation (2.4). Therefore, the only variable in the constraint (2.8) is the diameter of each link. As a result, the constraint (2.8) is the linear constraint for the model. The mathematical model is IP where the diameters $d_{ij} \in D$. Their solution technique is to relax the discrete variables of d_{ij} as continuous variables ($d_{ij} \geqslant 0$) which turns the IP model into the LP model. In this regard, they defined the weight for each arc (weight is the function of length and flow for each arc) in the network and converted the problem with the continuous variable d_{ij} into a problem with the continuous variable π_{ij}. Shiono and Suzuki (2016) showed that their solution methods do not require mathematical programming software. Also, the authors found the optimal continuous diameter without computing Lagrange multipliers. Shiono and Suzuki (2016) considered an iterative procedure that converts the original tree into a single equivalent arc (including two nodes: source node and a demand node), thereby helping in the computation of the cost

minimisation. The procedure to convert each path from root node 1 into a single equivalent arc (contraction procedure) is as follows:

Step 1: Find the parent of the deepest node (depth-first search) of the path from root node 1.

Step 2: Combine the children as a new node with new weight of arc.

Step 3: Remove the parent of the deepest child and connect the child to the grandparent with the new weight of arc.

Step 4: Continue Steps 1-3 until the parent is the root node 1 (the contracted tree includes two nodes: root node 1 and a child).

Step 5: Compute the minimum cost of the contracted tree.

Then, Shiono and Suzuki (2016) considered an iterative procedure to expand the contracted tree to the original tree to compute the optimal diameters as a continuous variable. Also, these authors provided an algorithm to convert the continuous diameter to the single diameter for each arc. Shiono and Suzuki (2016) demonstrated the efficiency of their algorithm (high accuracy in a short computational time) by testing of two sample networks. The first sample network consists of 19 arcs and 20 nodes (a supply node, 13 demand nodes and 6 junction nodes). The second sample network consists of 45 arcs and 46 nodes (a supply node, 22 demand nods and 23 junction nodes).

Mohajeri et al. (2012) presented a Mixed Integer Programming (MIP) model to minimise the total cost of the gas distribution network. The cost includes the establishing cost for Town Board Station (TBS) and the cost of gas transportation from TBS to consumers and among consumers. Mohajeri et al. (2012) assumed that the pressure at each node is not a variable and would not exceed from 30 psi. The variables, here, are the flow through each link and the diameter type for each link. However, these authors did not use the nonlinear equation (2.6) to find the variables of flows and diameters for each link. As Mohajeri et al. (2012) considered the tree structure, the flow through the element of the network can be easily determined by solving the equation (2.4) as explained earlier. Then, the diameter of links as the only variable has been chosen for the gas flow rate. For example, gas can flow through the pipe diameter of 63 mm when $f_{ij} \in$

[0,44] (m^3/h). Mohajeri et al. (2012) used an Ant Colony Optimisation (ACO) heuristic algorithm coded in MATLAB for solving the problem. The aim was to compare the performance of the exact method and heuristic method for this problem. Then, these authors provided different test problems using the ACO algorithm and the Branch and Bound (B & B) method as an exact method in CPLEX. Mohajeri et al. (2012) compared the results between the ACO algorithm and the B & B algorithm. The authors showed that the AOC method obtained a solution close to the B & B solution (exact optimal solution) with much less computation time. In addition, Mohajeri et al. (2012) tested the real case of the natural gas network in Mazandaran Gas Company in Iran using the ACO algorithm. Mohajeri et al. (2012) reported about 50 percent improvement of the AOC algorithm in the cost compared to the cost that has been estimated by Mazandaran Gas Company's experts.

de Mélo Duarte et al. (2006) developed the MINLP model to find the least cost combination of diameters from a discrete set of commercially available diameters for the pipes of a given gas network. de Mélo Duarte et al. (2006) used a Tabu search heuristic algorithm for the problem and then compared the results of their proposed algorithm with the result of a Genetic Algorithm (GA) method. de Mélo Duarte et al. (2006) tested both algorithms for 44 instances including 34 instances for small (up to 264 nodes) and medium (less than 1000 nodes) sized and 10 instances of large sized (more than 1000 nodes) problem. The performance of the GA algorithm was worse in most instances, in particular for the large size instances, it was not able to find a solution. As a result, de Mélo Duarte et al. (2006) showed that the Tabu search method is very promising for both quality of solution and computational time for all instances compared to the GA algorithm (de Mélo Duarte et al. (2006)).

Multi Diameter (MD)

In this case, a pipe contains different commercially available diameter segments that are serially connected. The multi diameters are suitable for long distances and provide a cheaper design compared with single diameter. The benefit of MD diameter over SD diameter, also, is that there are no integer variables in the mathematical model regarding the diameter type that leads to simplifying

the problem.

As noted earlier in SD section, apart from SD cases, Bhaskaran and Salzborn (1979) have also considered the variable δ_{ij}^d for MD of each link by adding the new constraints for each link (i, j) to the model as follows:

$$\begin{cases} \sum_{d \in D} \delta_{ij}^d = 1, & \text{the sum of fractions is 1} \\ \delta_{ij}^d \geq 0, & \text{for the case of multi diameter} \end{cases}$$

The constraint $\delta_{ij}^d \geq 0, \forall (i, j) \in A$ is used for the multi diameter. The Integer Programming model converts into the Linear Programming model. The model is implemented in APEX and tested with data from the Moomba gas field in South Cooper Basin in Australia.

Wu et al. (2007) presented an NLP model (Model 2 in Table 2.1) to minimise the cost of design of gas distribution network under steady state conditions. In particular, the decision variables are the length of pipes' diameter, the pressure at each node, and the flow through each link and its direction through the pipeline. The problem is a nonconvex and nonsmooth NLP problem due to the nonlinear, nonconvex and nonsmooth pressure drop constraints. Wu et al. (2007) presented a global approach based on Global Optimisation (GOP) primal-relaxed dual decomposition algorithm as a solution method. This method partitions the variables of the model into two groups. The iterative procedure of the GOP method solves each group of variables separately using primal problem and relaxed dual problem. The lower and upper bound of the optimal solution is provided by the primal part and the relaxed dual part respectively. The GOP method guarantees convergence to the global optimal for a kind of specific structure program in which the objective function and constraints must satisfy specific conditions. Wu et al. (2007) assumed that the direction of the flow through each link is not given. They defined $f_{ij}^2 = f_{ij}|f_{ij}|$ in the equation (2.7). The absolute value definition gives the variables of $f_{ij}^+ = max\{0, f_{ij}\}$ and $f_{ij}^- = max\{0, -f_{ij}\}$ and the equations $f_{ij} = f_{ij}^+ - f_{ij}^-$, $f_{ij}^+ \times f_{ij}^- = 0$, and $|f_{ij}| = f_{ij}^+ + f_{ij}^-$. This leads to the definition of $f_{ij}^2 = (f_{ij}^+)^2 - (f_{ij}^-)^2$ in the equation (2.7) in Model 2. Thus, Wu et al. (2007) converted Model 2 to the nonconvex quadratic model replacing new variables of f_{ij}^+ and f_{ij}^- for the flow and adding the new constraint of $f_{ij}^+ \times f_{ij}^- = 0$ to Model 2. These authors partitioned the variables of the converted model and developed

the GOP approach (Wu et al. (2007)). Also, Wu et al. (2007) tested their approach for two small size networks to show the performance of their algorithm. In an example of a network with 6 nodes, while the distances between nodes are 10 miles, the GOP algorithm presented a single pipeline diameter type for each link. However, for the network with 14 nodes while the distances between nodes are more 20 and less than 180 miles, the GOP algorithm presented two different diameters for two links and one type diameter for other links. There is no further details about their computational result.

Continuous Diameter (CD)

In the case of CD diameter, the pipe diameter is considered as continuous that should be obtained from an interval $[D_{min}, D_{max}]$. In the real world, there are the standard sizes of pipeline diameter provided by companies for the gas network design. Therefore, the design model with continuous diameter is reasonable if there is no standard size for the diameter. However, some researchers used the continuous diameter to overcome the difficulty of handling discrete variables of diameters as well as to convert the MINLP problem into the NLP problem.

De Wolf and Smeers (1996) presented, the NLP problem for the design and operation of gas transmission network while considering the nonlinear constraints of the gas law (equation (2.6)). De Wolf and Smeers (1996) are the earliest authors that provided a realistic model considering the nonsmooth, nonconvex equation (2.6) as the constraints in their proposed mathematical model. The problem was solved by a combination of Generalised Reduced Gradient and penalty methods. De Wolf and Smeers (1996) applied the methodology to a real-world situation of Belgian gas network with known structure from Norway to France. Considering this methodology, these authors showed the saving of 0.263 milion Belgian French per day. In the next section, we discuss the literature on designing of unknown tree network structure.

2.2.2 Unspecified Network

In this problem, the optimisation of network layout is coupled with pipe sizing analysis (Network design and Allocation problem). Therefore, all design variables f_{ij}, π_i, d_{ij}, and z_{ij} are considered which makes the problem difficult. Also,

the reliability of the demand nodes should be considered which is the number of independent paths from the source node to the demand nodes (level 1).

Some researchers have addressed the layout geometry optimisation of the pipe networks, neglecting the influence of the pipe sizing on layout determination. For example, there are some methods for designing the layout of a tree network without considering equation (2.6) such as spanning tree (Mohajeri et al. (2012)), and the shortest path (Tingzhe and Changgui (2005)).

To the best of our knowledge, there are no explicit developed mathematical models in the literature to optimise network layout and to determine the diameter type for each connected link (Network design and Allocation problem). In a given tree network, we presented Models 1, 2, and 3 in Table 2.1 when we consider three cases to determine the diameter of each link. For designing a tree network, by adding the connection links as the variables into the models shown in Table 2.1, the models will become difficult to solve. In this section, we consider the papers that proposed a method to improve the layout design for the models in Table 2.1. The researchers considered an initial design for the network structure and considered the allocation models in Table 2.1 to evaluate the objective function. Mostly, the researchers developed a two-stage iterative method. In the first stage, the optimal diameters for a tree network are determined using a nonlinear programming method. In the second stage, iteratively, the researchers remove a link from the network, then search to add a new link to the network aiming to find the minimum cost.

Single Diameter (SD)

The earliest work of this case was provided by Rothfarb et al. (1970). They presented a heuristic method to design the offshore natural gas for the given gas field and plant locations. In the first stage, an initial tree is generated based on experience. The diameter of the individual pipes is then computed using Model 1. In the second stage, local transformations called Δ-changes are applied as follows: a cycle is formed by adding an arc connecting a node to the closest non-adjacent one. Then another arc from this single cycle is deleted in turn and the corresponding flows, diameters, pressures, and costs computed. If an improved

feasible solution is found, one moves to it and iterates the procedure until no further improvements in the cost can be made by Δ-changes (Brimberg et al. (2003)).

Continuous Diameter (CD)

Recently, André et al. (2013) considered the problem of optimal design of a hydrogen transmission network. Since the authors considered designing of a hydrogen network for future use (particularly in France), there is no specific known diameters for hydrogen pipes. As the result, André et al. (2013) considered continuous diameter for their work ($D_{min} < d_{ij} < D_{max}$).

André et al. (2013) did not develop an explicit mathematical model to design the network layout and to determine the diameter type for each connected link (Network design and Allocation problem). They presented a heuristic method to improve the given tree network. It includes a two-stage method of optimal design and sizing of the hydrogen network. The outer level is to improve the tree network using Δ-change method that is similar to the proposed method by Rothfarb et al. (1970). The initial minimal length spanning tree has been generated based on the distances between nodes. Moreover, on the inner level, the NLP model (Model 3 in Table 2.1) for the specified tree network has been solved using the SNOPT solver (André et al. (2013)). The SNOPT is a software package for solving large-scale non-linear optimisation problems developed by Stanford University (Gill et al. (2005)). The proposed heuristic method by André et al. (2013) is described as follows:

Algorithm 1: Δ-change heuristic algorithm

Step 1: Determine the minimum length spanning tree as the initial topology.

Step 2: Determine the diameter for each link of the tree network (Model 3 in Table 2.1) using SNOPT solver.

Step 3: Sort the nodes for exploration either on a distance-to-source criterion or randomly. Select a subset of nodes or the entire set of nodes.

Step 4: For each selected exploration node i of the tree:

(a) Select the closest nodes j (in the Euclidean distance) not connected with an arc from the tree to i.

(b) Add the arc a_{ij} and determine the cycle so created.

(c) Remove one of the other arcs on the cycle in turn and evaluate the cost.

(d) As soon as the cost is improved (in Model 3), the new network replaces the previous best network.

Step 5: Repeat the previous steps until no further improvements of the cost are reached.

André et al. (2013) evaluated the quality of the solution of their Δ-change heuristic method according to the following strategies:

- For the small networks (7 nodes), they compared the Δ-change heuristic solution with the solution of the spanning tree algorithm. Also, André et al. (2013) compared the Δ-change heuristic solution with the solution of the Tabu search algorithm. From the numerical results, the following conclusions for two small networks can be presented:
  - Both the Δ-change and the Tabu search algorithms reduced the total cost of network compared to MST algorithm while the total length of the proposed network increased by more than 5 kilometres compared to the MST's length.
  - For both tests, the proposed network by the Δ-change algorithm reduced the cost significantly compared to the Tabu search's ones that are approximately 7% for test 1 and 18% for test 2.
- For the large networks (81 nodes), André et al. (2013) compared the Δ-change heuristic solution with the solution of the Tabu search algorithm

proposed by Brimberg et al. (2003). Also, the Tabu search proposed by Brimberg et al. (2003) was started with an initial minimum spanning tree network and using SNOPT solver for determining the diameter sizing. André et al. (2013) provided the tables to show the comparison between these two algorithms in terms of cost and computation time. As a result, these authors demonstrated that the Tabu search best solution was more expensive than Δ-change one.

André et al. (2013) also conducted the application of their Δ-change method to the case of development of the future hydrogen pipeline network in France.

Multi Diameter (MD)

To the best of our knowledge, there is no work for both developing a mathematical model or solution method for this part in any of the gas or water distribution networks. We will present our developed mathematical model considering the case of (MD) in the first section of the next chapter (Chapter 3). Our work falls into the following category that has been extracted from Figure 2.1.

Tree network → Unspecified network →
Network Design and Allocation Problem → Multi diameter(MD)

In the literature, the cases of single diameter and continuous diameter have been considered by Rothfarb et al. (1970) and André et al. (2013). There is no work in the literature that considers the case of multi diameter. Since different cases of diameter have different models, we will not be able to compare our work (the case of multi diameter) with the methods of the literature for the case of single diameter and continuous diameter for the Network design and Allocation Problem. For future work, providing that we change our multi diameter into a single or continuous diameter model, this will allow us to compare the performance of the three models for a specific data set.

In the next section, we present the literature on the gas and water distribution networks design for the cycle (loop) structure.

2.3 Cycle (Loop) Structure

The loop structure is a network when the gas is served from at least two directions to the users (level 2 reliability). Throughout these two directions, there is no arc in common. The main difficulty, here, is to analyse the flow through each link of the cycle due to the nonlinear equation (2.6). In order to solve the cycle network problem, it is prevalent to employ the two Kirchhoff's laws (Raoni et al. (2017)). The first (continuity equation) and second (energy conservation equation) laws were originally developed to solve electrical circuit problems. Raoni et al. (2017) presented these laws as follows:

The first Kirchhoff's law: This is a linear algebraic equation stating that the sum of the flows at the entrance to the node equals the sum of flows at the exit of nodes.

The second Kirchhoff's law: This is a nonlinear algebraic equation stating that the sum of the electrical potential differences around any closed circuit is zero.

In gas network, the first law is known as mass flow balance through the pipe that is equation (2.4), whilst the second law is known as the sum of pressure losses in any closed piping network loop (which equals zero and is expressed as equation (2.9) below). We consider O, a set of loops in the network with the elements of $\{o_1, o_2, o_3, ..., o_{|O|}\}$ ($|O|$ is the number of loops). According to Nasr and Connor (2014), the loop constraint is:

$$\sum_{(i,j)\in o} (\pi_i - \pi_j) = 0, \qquad \forall o \in O \tag{2.9}$$

In the next sections, we first present the literature on the Allocation Problem (Problem 2) in the given (specified) cycle (loop) network. Then, we present the literature on the Network Design and Allocation Problem (Problem 1) when the structure of the network is not given, but is required to be a loop.

2.3.1 Specified Network

There are two different approaches for the analysis of loop structure. The first approach is to analyse the equation systems of the cycle structure either for the gas or the water network. The second approach is to present mathematical modelling of the gas distribution network when the structure is cycle to minimise the design cost as well as determining the flow, the pressure and the diameter for each link. We explain in more detail below.

Analysis of the equation system of the cycle structure

The equations (2.4), (2.6), and (2.9) are considered as the equation system for the analysis of the cycle network (Krope et al. (2011)). We note that the network structure and the diameter of the pipeline are given, and the variables are just pressure and flow.

Many researchers worked on the analysis of the equation system of the cycle (loop) network to determine the pressure at each node of the loop as well as the flow through each link. The first relevant method to solve the nonlinear system of equation for the loop network (water or gas) proposed by Cross (1936) is primarily a relaxation method. In particular, the applied methods for this approach, are divided into three groups as detailed below, depending on which law should be first satisfied in the modelling (Raoni et al. (2017)).

- **Method of Balancing Flow:** In this method, the equation (2.4) would be satisfied. The Hard-Cross procedure proposed by Cross (1936) is a method to solve a nonlinear system of equations for the loop network with given pressures at any node of the network. The procedure includes the following steps:

  Step 1: Determine the flow through each link by solving equations (2.6) and (2.9).

  Step 2: Check if the flow satisfies equation (2.4), then stop, otherwise, go to Step 3.

  Step 3: Find of the pressure at the node and do Steps 1-2 until flow balance for each link is satisfied.

The correction of the node pressure in Step 3 above is needed to present new methods such as Simulated Annealing (Raoni et al. (2017)); Newton-Raphson (Sarbu (2014); Spiliotis and Tsakiris (2013)).

- **Method of Balancing Pressure:** In this method, the equation (2.6) is to be satisfied. The Newton-Raphson method is a method to solve problems when the inlet and outlet flow rates of the cycle network are known (Sarbu (2014); Spiliotis and Tsakiris (2013)). By satisfying the flow through each link of the loop, the pressure at each node is determined using equations (2.6) and (2.9).

- **The Hybrid Method:** This method does not need first to satisfy any Kirchhoff's laws, which leads to a cycle network problem characterised by both node mass balance and energy conservation equations (Agency (2014)). Notably, the EPANET software has been developed for simulation of the pipeline for the cycle water network (Agency (2014)).

The necessary input data for these methods are presenting the exact position of loops that means which links belong into which cycles. Today, the Hardy-Cross method (Cross (1936)) is very often used for analysis of cycle gas distribution networks. This method is powerful for the calculation of the cycle gas distribution network without limiting factors such as the number of nodes, the number of cycles, and the number of links as the input data (Brkić (2009)). We note the considerable number of commercial software packages developed in the market for the calculation of a cycle network in water distribution systems compared to gas networks (Raoni et al. (2017)).

Design of the cycle (loop) Distribution Networks

In this approach, the mathematical model nature of the design of the cycle (loop) gas and water distribution network is a MINLP problem in which the equation (2.9) for each cycle has been considered as a constraint.

Much attention has been given to the loop water distribution network rather than gas, as the later has been shown to be more complicated in practice. This can be a motivation for future work in the area of gas networks.

Table 2.2: Mathematical model for design of the cycle gas distribution network

		Objective function	Constraints
Model 4	Single Diameter(SD)	(2.1)	$(2.4)-(2.6),(2.9)$

Single Diameter (SD)

The early work for the design of the cycle (loop) water distribution network was by Alperovits and Shamir (1977) and Goulter (1992). The Water Distribution Network Design (WDND) mathematical model (similar to Model 4 in Table 2.2) is a NLP model to find the size of pipeline diameters from a discrete set, the pressure at each node and flows through each pipeline for a given loop water network structure that yields the least cost network and satisfied the constraints (2.4)-(2.9). The set of cycles (loops) O in Model 4 should be considered as an input data (Suribabu (2012)). De Corte and Sörensen (2013) presented a survey including the methods to solve the WDND problem (Model 4) and compared some of these methods providing the computational comparison results between them.

In addition, many researchers presented different methodologies for the NLP problem of Model 4 given in Table 2.2 (Allocation Problem), for the large problems. The techniques relevant to this problem includes: Genetic Algorithm (Haghighi et al. (2011)); Shuffled Frog Leap algorithm (Eusuff and Lansey (2003)); Honey-bee Mating optimisation (Suribabu (2012)). For large problems, Zheng et al. (2013) provided a novel optimisation approach namely a graph theory algorithm to identify the sub-networks for the original water network. The sub-networks, rather than the original water network, are individually optimised by a Differential Evolution (DE) method in a predetermined sequence. Zheng et al. (2013) tested their algorithm for five case studies to verify its effectiveness. The computational results also were compared to the results of standard DE (SDE) algorithm and Genetic Algorithm (GA). The results showed that their proposed method could find the same lowest cost solution for the small size problem while finding better feasible solutions for the larger size than the SDE and the GA methods.

2.3.2 Unspecified Network

In this section, the optimisation of the network layout is coupled with pipe sizing analysis. Therefore, all design variables f_{ij}, π_i, d_{ij}, and z_{ij} are considered which make the problem even more complicated. Also, the reliability of the demand nodes should be considered that is the two or more independent paths from the source node to each demand node (Afshar (2007)).

Some researchers, in this case, have addressed the layout geometry optimisation of the pipe networks, neglecting the influence of the pipe sizing on layout determination. Ting-zhe (2006) presented a mathematical model for the design of a loop network considering the supplying of each demand node from two directions. Ting-zhe (2006) did not consider the pipe sizing for the network; therefore, the equation (2.6) in their model was neglected. In contrast, these authors assumed the specific diameters according to the land-forms (a nature feature of the earth's surface) that lead to simplifying the model by considering only one type of variable (connection links).

To the best of our knowledge, there is no developed mathematical model in the literature to optimise the network layout and to determine the diameter for each connected link (Network design and Allocation problem). In Model 4, we define the allocation problem in the given loop networks. However, considering the connection links as the variables in Model 4 to design loop networks, make the problem even more difficult. As explained in Section 2.3.1, all proposed methods are presented to solve Model 4 (Allocation Problem). In Model 4, the network structure is given which is a cycle (loop) network. In this section, we consider the papers that proposed a method to improve the layout design as well as determining the diameters for each link in the given loop networks. In the literature, an initial design for the network is considered and Model 4 is used to evaluate the objective function. The standard approach is a two-stage iterative method. In the first stage, the optimal diameters for a cycle (loop) network (Model 4) are determined using a nonlinear programming method. In the second stage, an iterative pipe removal and the additional search process are carried out to reduce the cost (Afshar et al. (2005)). The second stage may lead to an infeasible solution due to undermining the node connectivity constraints or generating an infeasible network (Saleh and Tanyimboh (2013)).

Afshar et al. (2005) proposed for the second stage, a Floating method that has a significant effect on the optimality of the final solution. Possible removal of the pipes is decided upon by the optimisation algorithm assuming a zero value for the pipe diameter. By using this method, assuming a non-zero value for pipe diameter, it is highly possible for a removed pipe in a previous iteration to return into the network; provided its inclusion leads to a cheaper network in the current iteration. The methods for the second stage that have been proposed with an emphasis on presenting a feasible solution include the Min-Max Ant algorithm (Afshar et al. (2005)); a Genetic Algorithm using three modified roulette wheel selection schemes (Afshar (2007)); and the Conventional Roulette Wheel (Afshar (2007)).

In recent decades, the layout optimisation has focused, particularly on water distribution networks considering only single diameter for each link. However, more work needs to be done to optimise the network layout. In particular, this requires determining the diameter of each pipe in the case of multi diameter and continuous diameter for the gas or the water distribution networks.

2.4 The Research Gap and The Research Problem

To the best of our knowledge, there is no work for both developing a mathematical model or solution method for the case of multi diameter when the network is not given. Our work falls in the following category that has been extracted from Figure 2.1.

Tree network → Unspecified network →
Network Design and Allocation Problem → Multi diameter (MD)

The problem is:

- Given a set of nodes and customer requirements, determine the optimal network. This involves determining the network layout, the multi diameter of the connected pipeline, the pressure at each node, and the flow through

each link.

The objective is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which is a tree.

2.5 The Contributions of This Thesis

Our research provides a contribution to the areas of gas distribution network design and allocation. More specifically we contribute the following:

First contribution: Developed an effective mathematical model to design the tree network structure and to determine its components. These components includes the multi diameters for each link, the pressure at each node, and the flow through each link (Network Design and Allocation Problem). The objective is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which in our case is a tree. There are a few works for the fundamental problem in cases of Single and Continuous diameters. There is no work in the literature that considers the case of multi diameter.

Second contribution: Solved our MINLP model using an approximation method that handles the nonconvexities of the constraints. The algorithm that we use is the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method given by Viswanathan and Grossmann (1990). Our motivation to choose the OA/ER-/AP method among the other MINLP algorithms, is in its ability to handle the nonconvexity of the problem. This algorithm alternates between two sub-problems including the Non Linear Programming (NLP) sub-problem and the Master Mixed Integer Linear Programming (MILP) sub-problem. In the NLP sub-problem, the integer variables are fixed and the continuous variables are determined. In the master MILP sub-problem, the effect of nonconvexities is reduced by linearization of the nonlinear functions at the set of linearization points. This algorithm has not been used before to solve the MINLP problem in gas network area. Our contribution is that we review this algorithm and then examine its functionality and efficiency on our

MINLP model. We consider the effects of different parameters and stopping criteria in the levels of the NLP sub-problem and the Master MILP sub-problem to find a near optimal solution. Also, we presented the computational results for the OA/ER/AP algorithm considering different values of optimality gap for the Master MILP sub-problem. We acknowledge the single starting point and the multi starting points for the NLP sub-problems. Also, we discuss the comparative analysis between the NLP-Single start and the NLP-Multi start strategies considering different values of optimality gap for the Master MILP sub-problem. For some test cases, a decrease in the value of ROT(MILP), results in an increase in the computation time and an improvement in the total cost. As expected the NLP-Multi start yields an improved objective function value but requires more computational time than the NLP-Single start. The main outcome is a MINLP model that is effective for small sized networks.

Third contribution: Developed, implemented and tested a new heuristic algorithm to solve our model. We design the near optimal tree layout and determined its components (the multi diameter for each link, the pressure at each node, and the flow through each link). Our motivation is to develop a heuristic algorithm to solve large size networks within 4 hours. We also investigate the quantitative analysis of the effects of different parameters of our heuristic algorithm on optimal decisions. We compare the results in the objective function value and the computation time between the OA/ER/AP algorithm and our heuristic algorithm. The main outcome is a heuristic algorithm that finds a good quality feasible solution for the large sized test problems within 4 hours.

Chapter 3

Outer Approximation Algorithm

In this chapter, we develop an effective Mixed Integer Non-Linear Programming (MINLP) model for gas distribution network design and allocation (Fundamental Problem). The problem is to design the tree network structure and determine its components. These components includes determining the multi diameters for each link, the pressure at each node, and the flow through each link (Network Design and Allocation Problem). The objective is to meet demands at the minimum cost.

In our model, we also consider a wide range of design parameters including the number of demand nodes, the set of pipe diameter types as well as the length (distance) between nodes, and the pressure limits at each node. We also consider the constraints under steady state conditions such as pressure limits at each node, the flow balance through each link, the pressure drop equation for two ends of each link. The decision variables are the selected links connecting nodes, the flow through each link, the pressure at each node, and the selected multi diameters and their length for each link.

Our MINLP model has nonlinear and nonconvex constraints that makes our problem computationally difficult. We need a method to solve this MINLP model. The algorithm that we use is the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method developed by Viswanathan and Grossmann (1990). The advantage of the OA/ER/AP algorithm, over other MINLP algorithms, is its ability in handling the nonconvexity of the problem. In this chapter, we will present:

- Our new MINLP model including the notation and terminology.
- The Outer Approximation algorithm and its variations.
- The OA/ER/AP algorithm.

We also explain the effective choice of parameters and stopping criteria for the OA/ER/AP algorithm that facilitates the finding of a cost-effective, feasible solution in a reasonable amount of computation time (We allow a maximum 24 hours).

This chapter is organised as follows. We present our mathematical model to design the tree gas distribution networks in Section 3.1. The basic Outer Approximation (OA) algorithm is presented in Section 3.2 followed by a discussion of the variations of the OA algorithm in Section 3.3. The OA/ER and the OA/ER/AP algorithms that are extended from the OA algorithm are described in Section 3.4. In Section 3.5, we apply the OA/ER/AP method to our model through partitioning the variables and presenting the Master MILP sub-problem. The stopping criteria for the OA/ER/AP algorithm are explained in Section 3.6. We end this chapter with some concluding remarks in Section 3.7.

3.1 Mathematical Model

In this section, we develop a Mixed Integer Non Linear Programming (MINLP) model to design the layout of gas distribution networks and to determine the diameter for each link (Network Design and Allocation Problem). The network is modelled in terms of graph theory and consists of a set of nodes representing source supply and customer demand and arcs representing pipelines. In our problem, the required network has a tree structure which means the underlying graph is a tree. The structure is motivated by the cost consideration. We aim to determine the multi diameter types for each link (The case of Multi diameter (MD)). Our problem is:

- Given a set of nodes and customer requirements, determine the optimal network. This involves determining the network layout, the multi diameter of the connected pipeline, the pressure at each node, and the flow through each link.

The objective is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, gas pressure drop laws and network structure which is a tree (Fundamental Problem).

We consider some important features of the design of gas distribution systems based on the literature (Wu et al. (2007)) including:

- The pipeline diameter configuration must follow a descending order.
- There are no compressors and no nozzles in the network.
- All pipes in the network are of the same type of material.

3.1.1 Notation and Terminology

We consider a supply source node with multiple customers. The following notation and terminology is used in our mathematical model.

Sets:

$N = \{1\} \cup N_c$: The set of supply and demand nodes.
$N = \{1, 2, 3, \ldots, i, \ldots, n-1, n\}$.

N_c: The set of demand nodes $\{2, 3, \ldots, i, \ldots, n-1, n\}$;
obviously, $|N| = n = |N_c| + 1$.

D : The set of available pipe diameter types $\{d_1, d_2, \ldots, d_{|D|}\}$.

Parameters:

s_1 : The supply at source node 1.

s_i : The demand at consumer node i.

$C(d)$: The cost per unit length of pipe diameter type d.

L_{ij} : The length between node i and node j.

${\pi_i}^L$, ${\pi_i}^U$: The lower and upper square pressure limits at node i.

M : A large number.

Variables:

π_i : The square of gas pressure at node i.

f_{ij} : The flow rate from node i to node j.

z_{ij} : The binary variable indicating whether or not node i is serving node j;

$$z_{ij} = \begin{cases} 1, & \text{if } i \text{ is a serving node for } j \text{ (i} \rightarrow \text{j)}. \\ 0, & \text{otherwise.} \end{cases}$$

l_{ij}^d : The length of pipe diameter type d from node i to node j.

We consider if $l_{ij}^d = 0$, then there is no pipe diameter type d for link (i, j). We assume without loss of generality that the sum of demands should be equal to the supply of the source node.

$$s_1 = \sum_{i \in N_c} s_i \tag{3.1}$$

We present our developed mathematical model below. Since our model structure is not given, we need to add the variables z_{ij} into the known constraints

(2.4), (2.5) and (2.7) that are used in any mathematical model for the design of gas distribution networks with the given structure. The resulting model is given below.

Mathematical Models

The problem can be mathematically formulated as follows:

Model 5

$$\text{Minimise} \quad \sum_{i\in N}\sum_{j\in N}\sum_{d\in D} C(d) l_{ij}^{d} \tag{3.2}$$

$$\text{Subject to:} \quad \sum_{j\in N} z_{ij} f_{ij} - \sum_{j'\in N} z_{j'i} f_{j'i} = s_i, \qquad \forall i \in N \tag{3.3}$$

$$\sum_{i\in N} z_{ij} = 1, \qquad \forall j \in N_c \tag{3.4}$$

$$z_{ij}(\pi_i - \pi_j) = \beta \sum_{d\in D} l_{ij}^{d} d^{-5} f_{ij}^{2}, \qquad \forall i \in N, \forall j \in N \tag{3.5}$$

$$\sum_{d\in D} l_{ij}^{d} = z_{ij} L_{ij}, \qquad \forall i \in N, \forall j \in N \tag{3.6}$$

$$\pi_i{}^{L} < \pi_i < \pi_i{}^{U}, \qquad \forall i \in N \tag{3.7}$$

$$f_{ij} \geq 0, l_{ij}^{d} \geq 0, \qquad \forall i \in N, \forall j \in N, \forall d \in D \tag{3.8}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in N, \forall j \in N. \tag{3.9}$$

The objective function in (3.2) is to minimise the total cost corresponding to pipeline costs. Constraints (3.3) represent the flow balance through each node. The determination of the tree topology is expressed in constraint (3.4) which stipulates that each demand node has exactly one input connection link. The relationships between the mass flow rate through a pipe and its pressure value at endpoints are expressed by equation (3.5). Constraints (3.6) specify that whenever there is a link between two nodes ($z_{ij} = 1$), then, the summation of the length of pipelines with different diameters must be equal to the distance between two end nodes. Constraints (3.7) sets the lower and upper limits of the pressure square value at every node. The non-negativity (when $z_{ij} = 1$) of the variables is represented by constraint (3.8). The constraints (3.9) represent the binary variables of connection link between two nodes.

The constraint (3.3) is nonlinear because it is the product of two variables z_{ij} and f_{ij}. This constraint shows if there is a link between node i and j (when $z_{ij} = 1$) then the gas may flow through the link and the flow balance through the link should be considered. This formulation is computationally difficult to solve and so we now look towards simplification. We replace the nonlinear constraint with two linear constraints (3.10) and (3.11) below. The first linear constraint (Constraint (3.10)) expresses that if there is (is not) connection between node i and node j, then the gas can (cannot) flow through the link (i, j). The second linear constraint (Constraint (3.11)) gives the flow balance through each node.

$$f_{ij} \leq M z_{ij}, \qquad \forall i \in N, \forall j \in N \tag{3.10}$$

$$\sum_{j \in N} f_{ij} - \sum_{j' \in N} f_{j'i} = s_i, \qquad \forall i \in N. \tag{3.11}$$

The transformed mathematical model (Model 5) is as follows:

Model 6

$$\text{Minimise} \quad \sum_{i \in N} \sum_{j \in N} \sum_{d \in D} C(d) l_{ij}^d \tag{3.12}$$

$$\text{Subject to:} \quad \sum_{j \in N} f_{ij} - \sum_{j' \in N} f_{j'i} = s_i, \qquad \forall i \in N \tag{3.13}$$

$$f_{ij} \leq M z_{ij}, \qquad \forall i \in N, \forall j \in N \tag{3.14}$$

$$\sum_{j \in N} z_{ij} = 1, \qquad \forall i \in N_c \tag{3.15}$$

$$z_{ij}(\pi_i - \pi_j) = \beta \sum_{d \in D} l_{ij}^d d^{-5} f_{ij}^2, \qquad \forall i \in N, \forall j \in N \tag{3.16}$$

$$\sum_{d \in D} l_{ij}^d = z_{ij} L_{ij}, \qquad \forall i \in N, \forall j \in N \tag{3.17}$$

$${\pi_i}^L < \pi_i < {\pi_i}^U, \qquad \forall i \in N \tag{3.18}$$

$$f_{ij} \geq 0, l_{ij}^d \geq 0, \qquad \forall i \in N, \forall j \in N, \forall d \in D \tag{3.19}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in N, \forall j \in N. \tag{3.20}$$

This problem is a MINLP problem due to the nonconvex and nonlinear constraints (3.16) and the binary variables of connection link between two nodes. In this thesis, we develop new Model 5 and modify this to Model 6. Hereafter,

we consider Model 6 as our developed mathematical model. The model can be solved by an approximation algorithm which we detail in the next section.

3.2 Outer Approximation (OA) Algorithm

In this section, we first detail the basic form of a MINLP problem including initial conditions for variables and functions. Then, we provide the NLP subproblems and the Master MILP subproblem as the main components of the Outer Approximation (OA) algorithm Duran and Grossmann (1986). The OA algorithm and its variations are also presented.

Many optimisation problems are modelled as Mixed Integer Nonlinear Programming (MINLPs). Biegler and Grossmann (2004) provided a general classification of mathematical optimisation problems as well as a review of solution methods. They considered the major types of optimisation problems for continuous and discrete variable optimisation, particularly the NLP and the MINLP problems. Many algorithms have been proposed to solve convex MINLP problems. The main idea used in solution approaches of the existing algorithms for the MINLP problem, are linear approximations of the nonlinear functions (Geoffrion (1972); Duran and Grossmann (1986); Quesada and Grossmann (1992); Fletcher and Leyffer (1994); Grossmann and Kravanja (1995)). We first show the most basic form (algebraic form) of a MINLP problem below presented by Melo et al. (2014):

(P1)

$$\text{Minimise:} \qquad f(x,y) \tag{3.21}$$

$$\text{Subject to:} \qquad g_j(x,y) \leq 0, \qquad j \in J \tag{3.22}$$

$$x \in X, y \in Y \cap Z^{n_y}. \tag{3.23}$$

where,

- $f : R^{n_x+n_y} \rightarrow R$ and $g : R^{n_x+n_y} \rightarrow R^m$ are convex and differentiable functions.
- J is the index set of inequalities.

- x and y are the continuous and discrete variables, respectively.
- The set X is commonly assumed to be a convex compact set, e.g. $X = \{x|x \in R^n, Dx \leq d, x^L \leq x \leq x^U\}$.
- The discrete set Y corresponds to a polyhedral set of integer points, $Y = \{y|y \in Z^m, Ay \leq a\}$ which in many applications is restricted to 0-1 values, $y \in \{0,1\}^m$.

Mostly, the objective and constraint functions f, g have been considered linear in y. For example, Duran and Grossmann (1986) and Viswanathan and Grossmann (1990) considered these functions as follow:

$$f(x,y) = c^T y + r(x)$$

$$g(x,y) = By + q(x)$$

In terms of solution techniques for the MINLP problems, linear approximation algorithms have been presented by Biegler and Grossmann (2004). The basic element of linear approximation algorithms is a sequence of Mixed Integer Linear Programming (MILP) problem and NLP sub-problems. Each MILP problem is a relaxation of the MINLP problem and generates valid lower bounds on the optimal solution value of the original minimisation problem (Biegler and Grossmann (2004)).

The most successful algorithm (approximation algorithm) in the class of MINLP programming is the Outer Approximation (OA) algorithm that alternates between solving a MILP sub-problem and one or two NLP sub-problems. The main idea is to approximate problem (P1) by a MILP problem which is built with linearization of the objective, and the constraint functions of the problem (P1) at the set of linearization points $T = \{(x^0, y^0), (x^1, y^1), \ldots, (x^t, y^t)\}$ at each iteration. The components of the OA algorithm are the following sub-problems that provide upper and lower bounds along with updating points of the set T at each iteration.

3.2.1 The NLP Sub-problems

There are two NLP sub-problems as the main components of the OA algorithm for such a problem (P1).

NLP Sub-problem for Fixed y^t

$$(\text{NLP}(y^t)) \quad \begin{cases} \text{Minimise} & f(x, y^t) \\ \text{Subject to:} & g_j(x, y^t) \leqslant 0, \qquad j \in J \\ & x \in X. \end{cases}$$

The NLP sub-problem is built on the problem (P1) by fixing y at iteration t (y^t). We suppose the solution of the NLP(y^t) sub-problem is feasible with an optimal solution given by x^t. As the point (x^t, y^t) is a feasible solution of the problem (P1), it provides an upper bound, and this point is added to the set T. If the NLP(y^t) sub-problem is infeasible, then the following feasibility problem (NLPF(y^t)) is solved.

Feasibility Sub-problem for Fixed y^t

$$(\text{NLPF}(y^t)) \quad \begin{cases} \text{Minimise} & u \\ \text{Subject to:} & g_j(x, y^t) \leqslant u, \qquad j \in J \\ u \geq 0, \\ x \in X, u \in R^m. \end{cases}$$

The NLPF(y^t) sub-problem for the given y^t determines a continuous point that satisfies the equations and minimise the violation of the inequalities. We assume that (u, x^t) is an optimal solution of the NLPF(y^t) problem. The point (x^t, y^t) is added to the set T as the solution at iteration t. Therefore, the set T is updated with (x^t, y^t) from solving either the NLP(y^t) problem or the NLPF(y^t) problem.

3.2.2 The MILP Cutting Plane

The Master MILP sub-problem in the OA algorithm is the following M-MILP problem, which is built with linearization of the objective function and the con-

straint functions of the problem (P1) at the set of linearization points $T = \{(x^1, y^1), (x^2, y^2), \ldots, (x^t, y^t)\}$.

$$\text{(M-MILP)} \quad \begin{cases} \text{Minimise} \qquad \alpha \\ \text{Subject to :} \\ f(x^t, y^t) + \nabla f(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} \leq \alpha, \qquad \forall (x^t, y^t) \in T \\ g_j(x^t, y^t) + \nabla g_j(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} \leq 0, \quad j \in J, \forall (x^t, y^t) \in T \\ x \in X, y \in Y \cap Z^{n_y}, \alpha \in R^1. \end{cases}$$

where, $T = \{(x^i, y^i) | i \leqslant t\}$ and x^t is an optimal solution of the below problems.

$$\begin{cases} \text{NLP}(y^t), & \text{if the NLP}(y^t) \text{ subproblem is feasible.} \\ \text{NLPF}(y^t), & \text{if the NLP}(y^t) \text{ subproblem is infeasible.} \end{cases}$$

We assume that $(\alpha, x^{t+1}, y^{t+1})$ is obtained from the M-MILP problem. When only a subset of linearization is included, these commonly correspond to the violated constraints of the problem (P1). Alternatively, it is possible to include all linearization in the M-MILP problem. The solution of the M-MILP problem yields a valid Lower Bound ($LB = \alpha$) to the problem (P1). This bound is non decreasing with the number of linearization points T. We note that as the functions $f(x, y)$ and $g(x, y)$ are convex, the linearization in the M-MILP problem is the nonlinear feasible region in the problem (P1) (Biegler and Grossmann (2004)).

We present the basic OA algorithm that has been proposed by Duran and Grossmann (1986). Also, more details of this algorithm have been presented by Fletcher and Leyffer (1994); Melo et al. (2014). The algorithm starts using initialisation of Lower and Upper Bound (LB, UB), empty set T and fixed integer value of y^1. The first element of the set of T is $\{(x^1, y^1)\}$ where, x^1 is an optimal solution of the following sub-problems.

$$\begin{cases} \text{NLP}(y^1), & \text{if the NLP}(y^1) \text{ is feasible.} \\ \text{NLPF}(y^1), & \text{if the NLP}(y^1) \text{ is infeasible.} \end{cases}$$

To summarise the OA algorithm, we can state that the NLP sub-problems (NLP, NLPF) and the Master MILP (M-MILP) problem are solved successively in a cycle of iterations to generate the points (x^t, y^t). It terminates whenever the lower value exceeds the upper value, or the problem (M-MILP) is infeasible. The OA algorithm for such a problem (P1) is described as follows:

Algorithm 2: Outer Approximation (OA) Algorithm (Melo et al. (2014)).

Step 1: Set $LB = -\infty, UB = \infty, T = \emptyset, \varepsilon$: convergence tolerance. Select an integer $y^1 \in Z^{n_y}$ and set t= 1.

Step 2: Solve the NLP(y^t) sub-problem. One of the following cases must occur:

(a) Problem NLP(y^t) has a finite optimal solution (x^t, y^t) and objective value $f(x^t, y^t)$.

- Update the current upper bound estimate $UB = min\{UB, f(x^t, y^t)\}$
- If $UB = f(x^t, y^t)$, then $(x^*, y^*) = (x^t, y^t)$. ((x^*, y^*) is the optimal solution).
- Set $T = T \cup (x^t, y^t)$ and go to Step 3.

(b) Problem NLP(y^t) is infeasible then

- Solve problem NLPF(y^t) and obtain (x^t, y^t).
- Set $T = T \cup (x^t, y^t)$ and go to Step 3.

Step 3: Solve the current relaxation the Master integer M-MILP problem. If the M-MILP problem is feasible, then obtain $(\alpha, x^{t+1}, y^{t+1})$ and set $LB = \alpha$. One of the following cases must occur:

(a) If $UB - LB < \varepsilon$ or the M-MILP sub-problem is infeasible then Stop and report (x^*, y^*).

(b) If $UB - LB > \varepsilon$ then set $t = t + 1$ and go to Step 2 to test the algorithm for a new integer y^{t+1} as the solution obtained from the M-MILP sub-problem.

Figure 3.1 gives a flowchart of the OA algorithm for more clarification.

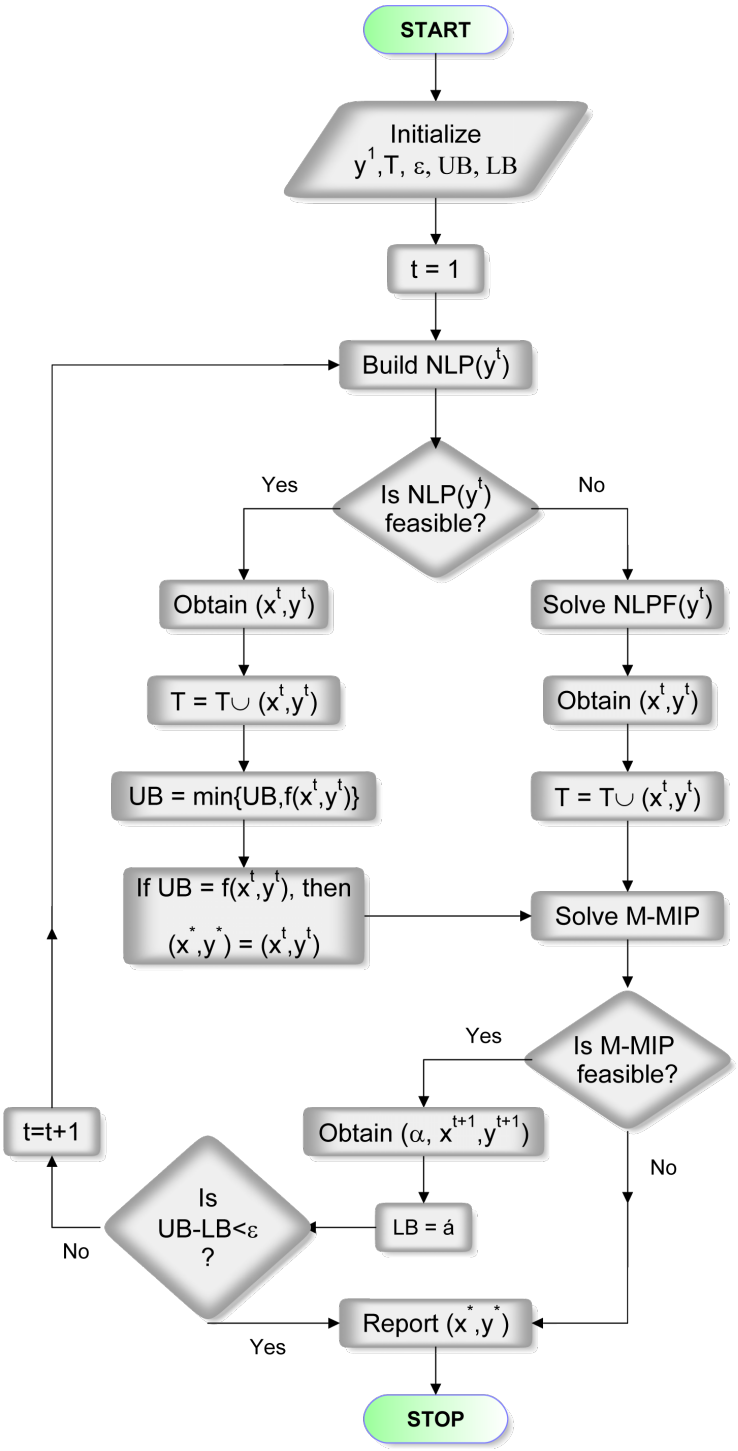

Figure 3.1: The Outer Approximation Algorithm

3.3 Variations of the OA Algorithm

In this section, we provide an overview of the main variations for the original OA algorithm that were proposed by Kocis and Grossmann (1987); Viswanathan and Grossmann (1990); Biegler and Grossmann (2004).

There are many MINLP problems which are not exactly in the form of the problem (P1). They may include linear and nonlinear equalities of the form $h(x, y) = 0$ or the convexity conditions may not hold for functions f, g and h. A good example that represents these additional conditions for a MINLP problem is our model (Model 6) given by equations (3.12)-(3.20) that includes the nonlinear and nonconvex equation (3.16). To explain the variations, we need to define the problem (P2) given by Kocis and Grossmann (1987) that is an extension of the problem (P1) and is as follows:

(P2)

$$\text{Minimise:} \quad f(x, y) \tag{3.24}$$

$$\text{Subject to:} \quad g_j(x, y) \leqslant 0, \qquad j \in J \tag{3.25}$$

$$h_l(x, y) = 0, \qquad l \in L \tag{3.26}$$

$$x \in X, y \in Y \cap Z^{n_y}. \tag{3.27}$$

where L is the index set of equalities and J is the index set of inequalities. The variables x are continuous and the variables y are discrete. The functions f, g, and h are defined over appropriate domains and have continuous partial derivatives. Also, the functions f, g, and h might be nonconvex in the problem (P2). We cannot solve the problem (P2) using the OA algorithm as the OA algorithm is not able to handle the equality constraints as well as the non-convexity of the problem in the Master MILP sub-problem. Therefore, we need to consider effective methods in the Master MILP sub-problem to overcome these difficulties (nonconvexity and equality constraints). The following methods are considered for the Master MILP problem to overcome the nonconvex and equality constraints h in the problem (P2).

- Addition of integer cuts when $y \in \{0, 1\}$,
- Handling of nonlinear equalities,

- Handling of nonconvexities.

We also consider an NLP sub-problem that is the Relaxed MINLP problem (P2). We provide the initial point (x^1, y^1) solving the Relaxed MINLP problem. The following methods can be applied for the NLP sub-problems of the OA algorithm.

- Initialise variable y^1 using the Relaxed MINLP problem,
- Multiple starting points for the NLP sub-problems.

We explain variations of the OA algorithm, first, at the level of Master MILP sub-problem and then, at the level of the NLP sub-problems.

3.3.1 In the Master MILP Sub-problem

In this section, we present the variations of the OA algorithm that we can use at the level of Master MILP sub-problem. The methods are presented below.

Integer Cut When $y \in \{0, 1\}$

One way to avoid solving the feasibility problem (NLPF) in the OA algorithm when the discrete variables in the problem (P1) and (P2) are $0-1$, is to introduce an integer cut. The aim is to make infeasible the choice of the previous $0-1$ values generated at all $t \in T$ previous iterations. This cut is defined as follows and given by Biegler and Grossmann (2004):

$$\sum_{i \in B^t} y_i - \sum_{i \in N^t} y_i \leq |B^t| - 1$$

where $B^t = \{i | y_i^t = 1\}, N^t = \{i | y_i^t = 0\}, t \in T$.

This cut becomes very weak as the dimensionality of the $0-1$ variables increases. However, it is useful feature in ensuring that new $0-1$ values are generated at each iteration (Duran and Grossmann (1986); Biegler and Grossmann (2004)).

Handling of Nonlinear Equalities

The OA algorithm (Algorithm 2) can solve the MINLP problem (P1) when it is limited to the only linear equality and linear (or nonlinear) inequality constraints. The OA algorithm is not able to handle the nonlinear equality constraints in the model. Some MINLP mathematical models contain nonlinear equality constraints like our model (Model 6).

For the case when linear equalities of the form $h(x, y) = 0$ are added to the problem (P1), there is no major difficulty since these are invariant to the linearization points (Biegler and Grossmann (2004)). If the equations are nonlinear, however, there are two difficulties. First, it is not possible to enforce the linearised equalities at all points of T. Second; the nonlinear equations may introduce nonconvexities unless they can be relaxed as convex inequalities (Biegler and Grossmann (2004)).

Kocis and Grossmann (1987) presented a new variant to the OA algorithm that can explicitly handle nonlinear equality constraints. They proposed an equality relaxation strategy in which the nonlinear equalities are replaced by the inequalities as follows:

$$\begin{cases} \nabla h_l(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} \leq 0, & \text{if } \lambda_l^t > 0 \\ -\nabla h_l(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} \leq 0, & \text{if } \lambda_l^t < 0 \end{cases} \tag{3.28}$$

where $(x^t, y^t) \in T, l \in L$ and λ_l^i is the Lagrange Multiplier associated with the equation $h_l(x, y) = 0$.

Handling of Nonconvexities

We consider the problem (P2) when the functions $f(x, y)$ and $g(x, y)$ are nonconvex or when we have nonlinear equalities $h(x, y) = 0$. The two difficulties arise. First, the NLP sub-problems (NLP) and (NLPF) may have multiple local

optimum solutions. Second, the Master problem (M-MILP) does not guarantee a valid lower bound LB or a valid bounding representation with which the global optimum may be cut off.

One general solution approach for handling nonconvexities, according to Biegler and Grossmann (2004), is to develop rigorous global optimisation algorithms, that assume specific forms of the nonlinearities (e.g. bilinear, linear fractional, and concave separable). The other option for handling nonconvexities is to apply a heuristic strategy to try to reduce as much as possible the effect of nonconvexities. While not being rigorous, this requires much less computational effort (Biegler and Grossmann (2004)). Viswanathan and Grossmann (1990) described the approach of reducing the impact of nonconvexities at the level of the Master MILP problem for the OA algorithm. They proposed a new Master MILP problem that incorporates an augmented penalty function for the violation of linearization of the nonlinear functions. The Master MILP problem reduces the likelihood of cutting-off feasible solutions (Viswanathan and Grossmann (1990)). This Master problem "Equality Relaxation/Augmented Penalty" (ER/AP) has the following form:

$$
\text{(M-ER/AP)} \quad
\begin{cases}
\text{Minimise} \qquad \alpha + \sum_{t\in T}[\omega_\rho^t \rho^t + \omega_\sigma^t \sigma^t] \\
\text{Subject to:} \\
f(x^t,y^t) + \nabla f(x^t,y^t) \begin{bmatrix} x-x^t \\ y-y^t \end{bmatrix} \leq \alpha, \\
\nabla h_l(x^t,y^t)^T \begin{bmatrix} x-x^t \\ y-y^t \end{bmatrix} \leq \rho^t, \qquad\qquad l \in L \\
-\nabla h_l(x^t,y^t)^T \begin{bmatrix} x-x^t \\ y-y^t \end{bmatrix} \leq \rho^t, \qquad\qquad l \in L \\
g_j(x^t,y^t) + \nabla g_j(x^t,y^t) \begin{bmatrix} x-x^t \\ y-y^t \end{bmatrix} \leqslant \sigma^t, \quad j \in J, \quad (x^t,y^t) \in T \\
\sum_{i\in B^t} y_i - \sum_{i\in N^t} y_i \leq |B^t| - 1, \qquad t \in T \\
x \in X, y \in Y \cap Z^{n_y}, \alpha \in R^1, \rho^t, \sigma^t \geqslant 0.
\end{cases}
$$

where $\omega_\rho^t, \omega_\sigma^t$ are weights that are chosen with sufficiently large values (e.g. 1000 times magnitude of Lagrange multiplier associated to the equations h_l and

g_j, respectively). We note that if the functions are convex, then all the slacks are set to zero on this Master MILP problem (M-ER/AP) that predicts rigorous lower bounds (Biegler and Grossmann (2004)).

3.3.2 In the NLP Sub-problems

In this section, we present the variations of the OA algorithm to find a good feasible solution at the level of the NLP sub-problem. The methods are presented below.

Initialise Variable y^1 Using the Relaxed MINLP Problem

As shown in the OA algorithm, the value of y^1 should be initialised by the user. This may not be a good choice. The method to estimate the initial value y^1 is to relax integer variables y of MINLP problem as continuous variables. The reason is the linear approximation to the MINLP problem at this point often leads to the good results (Viswanathan and Grossmann (1990)).

(RMINLP)

$$\text{Minimise:} \quad f(x,y) \tag{3.29}$$

$$\text{Subject to:} \quad g_j(x,y) \leqslant 0, \qquad j \in J \tag{3.30}$$

$$h_l(x,y) = 0, \qquad l \in L \tag{3.31}$$

$$x \in X, y \in Y_R. \tag{3.32}$$

where Y_R is the continuous relaxation of the set Y. The set T is initialised with the solution of the problem (RMINLP).

We note that when we try to overcome the effect of nonconvexities through the algorithm, it can fail to find the global optimum mainly for the two following reasons. Firstly, if the NLP sub-problem has multiple local solutions, then precisely the algorithm can converge to a sub-optimal point. Secondly, if the NLP sub-problem for fixed binary values has different local optima, the algorithm may be trapped into a local solution. In the next section, we explain that through

considering multiple start points, it may provide a better chance of finding a cost-effective feasible solution.

Multiple Starting Points for the NLP Sub-problems

As the NLP sub-problem has multiple local solutions, the algorithm is not guaranteed to find the global optimum. Practical experience has shown that it is sometimes difficult to get a feasible or better solution to the NLP sub-problems, in particular, for the initial point (x^1, y^1) (Kan and Timmer (1987)). Therefore, if we consider the widely spaced starting points for the NLP problem, then there may be a much better chance of finding a good feasible solution. We first present the Multi start algorithm for Unconstrained Programming that was proposed by Kan and Timmer (1987); Ugray et al. (2007). Then, we present the Multi start algorithm for constraint programming that was presented by Bisschop and Roelofs (2006).

- **Multiple Starting Points for Unconstrained Programming**

The earlier work focused on unconstrained problems where there are no discrete variables (Kan and Timmer (1987)):

(UCP)

$$\text{Minimise:} \qquad f(x) \tag{3.33}$$

$$\text{Subject to:} \qquad x \in X. \tag{3.34}$$

where $f : X \subset R^n \longrightarrow R$, and all global minima of f assumed to occur in the interior of X. The multi start algorithm is a stochastic method that attempts to find the global solution. Most stochastic methods include two phases (Kan et al. (1985)):

- **Global phase:** In this phase, the points are sampled randomly from a uniform distribution over X.
- **Local search:** In this phase, the sample points are controlled by doing of local searches to yield a candidate global minimum.

The most basic multi start algorithm presented by Ugray et al. (2007), is as follows:

1. A uniformly distributed sample of N_s points in X is generated,
2. The objective f is evaluated at each point. The points are sorted according to their f values, and the γN_s best points are retained, where γ is an algorithm parameter between 0 and 1.
3. L is started from each point of this reduced sample, except if there is another sample points within a certain critical distance that has a lower f value. L is also not started from sample points that are too near the boundary of X, or too close to a previously discovered local minimum.
4. N_s additional uniformly distributed points are generated, and the procedure is applied to the union of these points and those retained from previous iterations. The critical distance referred to above decreases each time a new set of sample points is added.

This also makes choosing the sample size N_s important, since too small sample leads to many revised decisions, while too large sample will cause L to be started many times (Ugray et al. (2007)).

- **Handling of Constraints for Constrained Programming**

Most classical search and optimisation methods handle constraints of the NLP problems by using a penalty function, where infeasible solutions are penalised depending on the amount of constraint violation (Deb and Agrawal (1999)). The following simple procedure is presented for the handling of constraints in the NLP sub-problems using penalty function. Here, we apply this procedure, in particular, for the Relaxed NLP sub-problem.

(CP)

$$\text{Minimise:} \qquad P(x,y,w,r) = f(x,y) + \sum_{j\in J} w_j (max(0, g_j(x)))^2 + \sum_{l\in L} r_l [h_l(x,y)]^2 \tag{3.35}$$

$$\text{Subject to:} \qquad x \in X, y \in Y_R. \tag{3.36}$$

where the function $max(0, g_j(x))$ is the absolute amount by which the jth constraint is violated at the point (x,y). The w_j is a non-negative penalty weight of the jth inequality constraint. The r_l is the penalty parameter of the lth equality

constraint (Deb and Agrawal (1999)).

The penalised function $P(x, y, w, r)$ makes the constraints of the NLP sub-problem into an unconstrained minimisation problem. In addition, the optimal solution of the NLP sub-problem also is optimal for its (CP) problem. Furthermore, an optimal solution of the (CP) problem will provide an upper bound on the optimum for the NLP sub-problem, and this bound, in general, is tighter than that obtained by simply optimising the NLP problem (Smith and Coit (1997)). As all NLP sub-problems of the OA algorithm are constraint programming, the reduced sample is evaluated base on the penalised objective values.

Many local searches of the basic multi start algorithm for unconstrained programming, may converge to the same local minimum. To reduce computational time, Bisschop and Roelofs (2006) presented a multi start algorithm using clustering method (Clustering is a process of partitioning a set of data into a set of groups). These clusters are updated (and become larger) whenever a starting point is found that leads to a local solution that has already been found before. We present this algorithm below and call it the Multi-start algorithm (Algorithm 3). The inputs for Algorithm 3 are the values of (N_s) and (γN_s):

- Number of Sample Points (N_s): This is the number of randomly generated points in each iteration from the uniform distribution. The sample point is randomly generated by using the intervals defined by the lower and upper bounds of the variables. If a variable has no upper bound then the upper bound used for generating sample points will be equal to the value of this parameter. If a variable has no lower bound, then the value of this parameter multiplied by -1 will be used as a lower bound (Bisschop and Roelofs (2006)).

- Number of Selected Sample Points (γN_s): This is the number of points that are selected from the set of randomly generated points in each iteration, where the γ is a number between 0 and 1 (Bisschop and Roelofs (2006)). It is clear that $\gamma N_s \leqslant N_s$. As explained earlier, our NLPs sub-problems are the NLP constraint programming. Therefore, the selected points are the points with the best penalised objective value. In particular, if $N_s = \gamma N_s$ then all sample points are automatically selected (Bisschop and Roelofs (2006)).

In the computational results, we experiment with these two numbers with different sized test problems to find a good feasible solution.

Algorithm 3: Multi Start Algorithm for the NLP problem (Bisschop and Roelofs (2006))

Step 0: Set iteration counter equal to 1.

Step 1: Generate N_s sample points from a uniform distribution. then, group them into clusters. Calculate the penalised objective for all sample points and select the best γN_s sample points.

Step 2: For all the best sample points (γN_s) do:

- For all clusters, calculate the distance between the sample point and the centre of the cluster. If the distance is smaller than the radius of the cluster (i.e., the sample point belongs to the cluster) then delete the sample point.

Step 3: For all (remaining) sample points do:

- Solve the NLP problem by using the sample point as its starting point to obtain a candidate local solution.
- For all clusters do:
  - Calculate the distance between the candidate local solution and the local solution belonging to the cluster.
  - If the distance equals 0 (which implies that the candidate local solution is the same as the local solution belonging to the cluster) then update the centre and radius of the cluster by using the sample point.
  - Else, construct a new cluster by using the mean of the sample point and the candidate local solution as its centre with the radius equal to half the distance between these two points. Assign the candidate local solution as the local solution belonging to the cluster.
  - For all remaining sample points, calculate the distance between the sample point and the centre of the updated or the new cluster. If the distance is smaller than the radius of the cluster, then delete the sample point.

Step 4: Increment iteration count. If the number of iterations exceeds the iteration limit, then go to step (5). Else go to Step 1.

Step 5: Order the local solutions and store the number of best solutions in the solution repository.

3.4 Extended Algorithms of Outer Approximation Algorithm

The first extension algorithm of the OA algorithm is Outer Approximation/Equality Relaxation (OA/ER) that has been proposed by (Kocis and Grossmann (1987)), where convex MINLP problem (P1) is considered with additional equality constraints. The Outer Approximation algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method is an extension of both algorithms the OA algorithm and the OA/ER algorithm that has been proposed by Viswanathan and Grossmann (1990). The (OA/ER/AP) algorithm solves the nonconvex MINLP problem such as the problem (P2).

Outer Approximation/Equality Relaxation (OA/ER) Algorithm

Kocis and Grossmann (1987) proposed a method to solve such a MINLP problem (P2) under the assumption of convexity of the functions f, g and the quasi-convexity of non linear equality constraints. The proposed (OA/ER) method is the earliest work for the MINLP problem with particular attention given to the handling of nonlinear equality constraints.

The (AP/ER) algorithm are different to the OA algorithm (Algorithm 2) in some steps. In the OA/ER algorithm, a new Master MILP problem is defined in a way that nonlinear equations in the MINLP formulation can be handled explicitly. Thus, compared with the OA algorithm, two different steps are presented as follows:

Step 1: The initial value for the set T as starting point is obtained from Relaxed MINLP problem.

Step 3: The Master MILP is generated using equality relaxation strategy of equation (3.28) which has been presented in the earlier sections.

Outer Approximation Algorithm/Equality Relaxation/Augmented-Penalty (OA/ER/AP) Algorithm

Viswanathan and Grossmann (1990) improved the OA algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method to solve a MINLP problem (P2) where there are equality constraints h and the convexity conditions may not hold for functions f, g, and h. The method of OA/ER/AP for handling the nonlinear equality is based on equality relaxation strategy which is proposed in the OA/ER algorithm. Then, the proposed Master MILP problem (M-ER/AP) uses a linear approximation to an exact penalty function that allows violations in the linearization of the nonlinear functions (Viswanathan and Grossmann (1990)). As shown in the earlier sections (Section 3.3), the Master (M-ER/AP) problem can overcome the nonconvexity of constraints (Equation (3.26)) of the problem (P2).

The proposed algorithm by Viswanathan and Grossmann (1990) involves the same steps as the OA algorithm, but the set T is initialised by solving the Relaxed MINLP problem. If an integer solution is not found, a sequence of iterations consisting of the NLP sub-problems and the Master MILP problem (M-ER/AP) are solved. The search proceeds until no improvement are found in the NLP sub-problems (Viswanathan and Grossmann (1990)). The following OA/ER/AP algorithm has been presented by Bisschop and Roelofs (2006).

Algorithm 4: Outer Approximation/Equality Relaxation/Augmented Penalty (OA/ER/AP) algorithm (Bisschop and Roelofs (2006)).

Step 1: First, all the integer variables are relaxed as continuous variables between their bounds for the model. The nonlinear sub-problem (RMINLP) is solved using any nonlinear solver.

Step 2: Then a linearization is carried out around the optimal solution, and the resulting constraints are added to the linear model already presented. This new linear model is referred to as the Master MILP model.

Step 3: The Master MILP model is solved using any MILP solver.

Step 4: The integer part of the resulting optimal solution is then temporarily fixed, and the original MINLP model with fixed integer variables is solved as a nonlinear sub-model using the NLP solvers.

Step 5: Again, a linearization around the optimal solution is constructed, and the new linear constraints are added to the Master MILP model. To prevent cycling, one or more constraints are added to cut off the previously found integer solution of the Master model.

Step 6: Steps 3-5 are repeated until one of the termination criteria (will be discussed further in Section 3.6) is satisfied.

3.5 The OA/ER/AP Implementation for the Gas Distribution Network (Model 6)

In this section, we represent the implementation of the OA/ER/AP algorithm for our MINLP model developed for the gas distribution network problem (Model 6) given by equations (3.12)-(3.20). We partition the variables into two subsets: continuous and binary variables. The continuous variables consist of $f_{ij}, \pi_i,$ and l^d_{ij}, and binary variables are z_{ij}. We define the following set of continuous variables X and set of binary variables Y for our problem (Model 6).

$$x \equiv \{l^d_{ij}, f_{ij}, \pi_i\}, \quad X = \{x | \pi^L_i \leq \pi_i \leq \pi^U_i, l^d_{ij}, f_{ij} \geq 0, \ \forall i, j \in N\}$$

$$y \equiv \{z_{ij}\} \in R^n, \ \ Y = \{y | z_{ij} \in \{0, 1\}, \ \forall i, j \in N\}$$

We also need to initialise the value of $LB = -\infty, UB = \infty, T = \emptyset$. We define ε, as the convergence tolerance.

Step 1: We relax the binary variables y where $Y_R = \{y | 0 \leqslant z_{ij} \leqslant 1\}$ for

our model (Model 6) and replace the Constraint (3.20) by the Constraint $0 \leqslant z_{ij} \leqslant 1$. The relaxed MINLP problem can be solved using any nonlinear solver and the results are (x^1, y^1) and $T = \{(x^1, y^1)\}$. If the relaxed MINLP problem is infeasible, then we solve the feasibility sub-problem (mentioned earlier in Section 3.2.1) for $0 \leqslant z_{ij} \leqslant 1$.

Step 2: In this step, the nonlinear part of our model is linearized. The linearization is defined as a linear approximation of the nonlinear equation that is valid in a small region around an operating point. In this algorithm, the linearization is implemented around the points of set T which include the optimal solution point that is obtained from all previous steps. The aim is to build the Master MILP model for our MINLP model. We note that, in our mathematical model, the equation (3.16) is the only nonlinear equation. This equation is also nonconvex. Therefore, for the definition of the Master MILP problem, we need to transform the equation (3.16) to an inequality like function (3.28). We consider the following equation,

$$h_{(i,j)} = z_{ij}(\pi_i - \pi_j) - \beta(\sum_{d \in D} l_{ij}^d d^{-5}) f_{ij}^2 \ , \qquad \forall i \in N, j \in N$$

We define the equation (3.37) by:

$$\nabla h_{(i,j)}(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} \leq 0 \tag{3.37}$$

where,

$$\nabla h_{(i,j)}(x^t, y^t) \begin{bmatrix} x - x^t \\ y - y^t \end{bmatrix} = z_{ij}^t((\pi_i - \pi_j) - (\pi_i^t - \pi_j^t)) + (\pi_i^t - \pi_j^t)(z_{ij} - z_{ij}^t) -$$
$$\beta \sum_{d \in D} (l_{ij}^d - (l_{ij}^d)^t) d^{-5} (f_{ij}^t)^2 - 2\beta (f_{ij})^t \sum_{d \in D} (l_{ij}^d)^t d^{-5} (f_{ij} - f_{ij}^t) = z_{ij}^t(\pi_i - \pi_j) +$$
$$(\pi_i^t - \pi_j^t)(z_{ij} - 2z_{ij}^t) - \beta f_{ij}^t [\sum_{d \in D} l_{ij}^d d^{-5} + \sum_{d \in D} (l_{ij}^d)^t d^{-5} (f_{ij}^t - 2f_{ij})],$$
$$\forall i \in N, j \in N \qquad \forall t \in T. \tag{3.38}$$

As explained in earlier Section 3.3, the equalities and nonconvexities of the constraints are handled in the level of the Master MILP (M-ER/AP) problem. The Master MILP problem for our model (Model 6), then, is built

as follows for all $(x^t, y^t) \in T$ including equations (3.38). The Master MILP sub-problem for our model is satisfied the OA/ER/AP algorithm.

M-Model 6

Minimise:

$$\alpha = \sum_{i\in N}\sum_{j\in N}\sum_{d\in D} C(d) l_{ij}^d + \sum_{t\in T} \omega_\rho^t \rho^t \tag{3.39}$$

Subject to:

$$\sum_{j\in N} f_{ij} - \sum_{j'\in N} f_{j'i} - s_i \leq 0, \qquad \forall i \in N \tag{3.40}$$

$$-\sum_{j\in N} f_{ij} + \sum_{j'\in N} f_{j'i} + s_i \leq 0, \qquad \forall i \in N \tag{3.41}$$

$$f_{ij} - M z_{ij} \leq 0, \quad \forall i \in N, \forall j \in N \tag{3.42}$$

$$\sum_{i\in N} z_{ij} - 1 \leq 0, \qquad \forall j \in N_c \tag{3.43}$$

$$-\sum_{i\in N} z_{ij} + 1 \leq 0, \qquad \forall j \in N_c \tag{3.44}$$

$$\begin{aligned} & z_{ij}^t(\pi_i - \pi_j) + (\pi_i^t - \pi_j^t)(z_{ij} - 2z_{ij}^t) \\ & - \beta f_{ij}^t [\sum_{d\in D} d^{-5} l_{ij}^d + \sum_{k\in D} d^{-5} (l_{ij}^d)^t (f_{ij}^t - 2f_{ij})] \leq \rho^t, \\ & \qquad \forall i \in N, \forall j \in N, \forall t \in T \end{aligned} \tag{3.45}$$

$$\begin{aligned} & -[z_{ij}^t(\pi_i - \pi_j) + (\pi_i^t - \pi_j^t)(z_{ij} - 2z_{ij}^t) \\ & - \beta f_{ij}^t [\sum_{d\in D} d^{-5} l_{ij}^d + \sum_{d\in D} d^{-5} (l_{ij}^d)^t (f_{ij}^t - 2f_{ij})]] \leq \rho^t, \\ & \qquad \forall i \in N, \forall j \in N, \forall t \in T \end{aligned} \tag{3.46}$$

$$\sum_{d\in D} l_{ij}^d - z_{ij} L_{ij} \leq 0, \qquad \forall i \in N, \forall j \in N \tag{3.47}$$

$$-\sum_{d\in D} l_{ij}^d + z_{ij} L_{ij} \leq 0, \qquad \forall i \in N, \forall j \in N \tag{3.48}$$

$$\sum_{(i,j)\in B^t} z_{ij} - \sum_{(i,j)\in N^t} z_{ij} \leq |B^t| - 1, \qquad \forall t \in T \tag{3.49}$$

$$\underline{\pi_i} < \pi_i < \bar{\pi}_i, \qquad \forall i \in N \tag{3.50}$$

$$f_{ij} \geq 0, l_{ij}^d \geq 0, \rho \geq 0, \qquad \forall i \in N, \forall j \in N \tag{3.51}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in N, \forall j \in N. \tag{3.52}$$

where, $\omega_\rho^t > |\lambda_{(i,j)}^t|$ is the weight on the slack variables ρ with λ_{ij}^t being

the Lagrange multiplier of equation (3.16) for each $i \in N, j \in N$ and $B^t = \{(i,j)|z_{ij}^t = 1\}, N^t = \{(i,j)|z_{ij}^t = 0\}$.

The MILP problem (M-Model 6) is solved using any MILP solver. $x^t = \{\pi^t, f_{ij}^t, (l_{ij}^d)^t\}, y^t = A^t$. Also, the results are $\alpha = LB$,
$x^{t+1} = \{\pi^{t+1}, f_{ij}^{t+1}, (l_{ij}^d)^{t+1}\}, y^{t+1} = A^{t+1}$.

Step 3: In this step, our MINLP problem (Model 6) is transformed to the NLP problem where the integer variables $y^{t+1} = A^{t+1}$ are fixed. The integer variables are the solution of the (M-Model 6) problem as is explained in Step 2. It provides the Upper Bound UB as well as the variables of x^{t+1}. If the NLP(y^{t+1}) problem is infeasible, then the NLPF(y^{t+1}) problem will be solved. We set $T^{t+1} = \{(x^1, y^1), (x^2, y^2), (x^3, y^3), \ldots, (x^{t+1}, y^{t+1})\}, t \in T$.

Step 4: The relation between the Master MILP (M-Model 6) problem and the NLP problems will continue until the Master MILP problem becomes infeasible or $UB - LB < \varepsilon$.

3.6 Effective Parameters and Stopping Criteria for the OA/ER/AP Algorithm

The following stopping criteria can be considered for the OA/ER/AP algorithm.

- **Iteration limit:** Limit the number of iterations the OA/ER/AP algorithm is allowed to perform which is normally 20 iterations. There are two reasons for using an iteration (Bisschop and Roelofs (2006)):
  - A good solution is usually found during the first few iterations.
  - The size of the underlying the Master MILP model tends to grow significantly each time linearization constraints are added, causing an increase in computation time.
- **Objective value worsening:** A second criterion is the worsening of the objective function value ($f(x^t, y^t)$) of two feasible nonlinear sub-models of the NLP(y^t), the NLP(y^{t+1}). The underlying reason is that the Master MIP model will not always select binary solutions that lead to successively

improving NLPs. This criterion seems appropriate when the worsening is persistent over several iterations (Bisschop and Roelofs (2006)).

- **Small Optimality Gap:** A third termination criterion is an insufficient improvement in the objective function value of the Master MIP model (LB) about the objective function value of the previously solved the NLP(y^t) sub-problem (UB). The difference between these two values represents the optimality gap since the Master MILP model represents a relaxation of the original MINLP model.

In particular, there is a stopping criterion for the Master MILP (M-Model 6) problem. The iteration procedure for solving the MILP problem is to relax the integer variables and then solve the resulting linear problem. The relaxation of our Master MILP (M-Model 6) problem provides the lower bound (RLB) for our Master MILP problem. The upper bound here is the objective function value of the previously solved the NLP sub-problem. We define the Relative Optimality Tolerance to guarantee that the value of RLB lies within a certain percentage of the value of UB. Sometimes the Master MILP problem finds a good integer solution early but must examine many additional nodes to prove the solution is optimal. We can speed up the process changing the optimality tolerance (Bisschop and Roelofs (2006)). Therefore, we can experiment with the values of following stopping criterion for solving the Master MILP problem.

- **MILP Relative Optimality Tolerance ROT(MILP):** The solution procedure stops if the solver can guarantee that the current best solution is within 100 $\times$ ROT(MILP) of the global optimum (UB-RLB$<$ 100$\times$ ROT(MILP)).

3.7 Conclusion

We developed a Mixed Integer Non Linear Programming (MINLP) model to design of the tree gas distribution networks. We considered a wide range of design parameters including the number of supply and demand nodes, the set of pipe diameter types, the length (distance) between nodes, and the pressure limits at each node. We consider the constraints under steady-state conditions such as pressure limits at each node, the flow balance through each link, the pressure drop equation for two ends of each link. The decision variables are the selected

links connecting nodes, the flow through each link, the pressure at each node, and the length of selected diameter for each link.

Our model is a MINLP problem with nonconvex constraints. We aimed to solve our model using the Outer Approximation algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method given by Viswanathan and Grossmann (1990). The advantage of the OA/ER/AP algorithm, among other MINLP algorithms, is its ability in handling the nonconvexity of the problem. The OA/ER/AP method is an extension of the OA algorithm. We presented an overview of the OA algorithm and its variations. The OA/ER/AP algorithm alternates between two sub-problems; the NLP sub-problems; and the Master MILP sub-problem. The detailed algorithm is presented, and the effects of different parameters and stopping criteria in the levels of the NLP sub-problems and the Master MILP sub-problem on optimal decisions are also discussed. To implement this algorithm to solve our MINLP model, we partitioned the variables into two subsets of variables: the continuous and the binary variables. Also, we developed the Master MILP sub-problem for our model. We detailed how the equalities and nonconvexities of the constraints are handled in the level of the Master MILP (M-ER/AP) problem.

Chapter 4

Computational Results

In this chapter, we test our MINLP model (Model 6) given by equations (3.12)-(3.20) using the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method (Algorithm 4 detailed in Chapter 3). This algorithm alternates between two sub-problems including the NLP sub-problems and the Master MILP sub-problem. In the NLP sub-problems, the integer variables of our model are fixed (z_{ij}) and the continuous variables are determined. In the master MILP sub-problem, the effect of nonconvexities of our model are reduced by linearization of the nonlinear functions of the model at the set of linearization points at each iteration. We presented the master MILP model (M-Model 6) given by equations (3.39)-(3.51) for our model (Model 6) in Chapter 3. The NLP sub-problems and MILP provide upper and lower bounds along with updating the set of linearization points at each iteration.

The aim of the computational results is to validate the model and demonstrate that a cost-effective feasible solution can be generated in a reasonable amount of computational time (24 hours) for various test cases. We note that, for the the OA/ER/AP algorithm, we allow up to 24 hours of computation time for each instance. We generate different network sizes (5 to 50 nodes) to test our model with 30 instances for each size. The computational results are undertaken considering different values of ROT(MILP) for the Master MILP sub-problem and under the conditions of Single start and Multi start for the NLP sub-problems.

This Chapter is organised as follows. In Section 4.1, we detailed how we generate the test data for our model. We present numerical results of different

test cases with a Single start for the NLP sub-problems and different optimality gap values of the Master MILP sub-problem (ROT(MILP)) in Sections 4.2. In Section 4.3, we provide the numerical results of different test cases with Multi start for the NLP sub-problems and different optimality gap values of the Master MILP sub-problem (ROT(MILP)). The comparative analysis between the NLP-Single start and the NLP-Multi start strategies are discussed in Section 4.4. We finish this chapter with some concluding remarks in Section 4.5.

4.1 Generating Test Data

We presented the input parameters for our mathematical model in Section 3.1.1. We detail below how we generate the test data for our model. There is very little test data in the literature for our problem. In the literature, there are a few papers on the Allocation problem which provide test data for small sized networks. Whilst our data is not extracted from the literature, we adopted some ideas from Mohajeri et al. (2012) and Wu et al. (2007) on how to generate the test data. More details of generating the test data are given below. We also need to note that, we may not be able to compare our work to Mohajeri et al. (2012) and Wu et al. (2007) as they worked on the sub-problem (Allocation Problem) and our work is focused on the fundamental problem (Network Design and Allocation Problem).

- s_i (The demand at node i): We generate randomly three different types of demand values for consumers as Low, Medium and High demands for the network with 49 demand nodes. These different demand numbers are evaluated as follows:

  - Low demand: $s_i \in [200, 450]$
  - Medium demand: $s_i \in (450, 1000]$
  - High demand: $s_i \in [800, 1500]$

  $$\{s_2, \quad s_3, \quad s_4, \quad s_5, \quad s_6, \ldots, s_i, \ldots, \quad s_{49}, \quad s_{50}\}$$

  The amount of supply at, s_1, the source node is set as the sum of demands, in other words $s_1 = \sum_{i=2}^{50} s_i$.

Our test data is for $n = 5$, 10, 15, 20, 25, and 50. The demand data for values of $n \leq 25$ is obtained from the $n = 50$ data by taking the elements $\{s_2, s_3, s_4, s_5, s_6, \ldots, s_n\}$. For example, for the category of 5 nodes, we select the four first of values from the network with 49 nodes as shown below.

$$\{\{s_2, \quad s_3, \quad s_4, \quad s_5 \quad\}, s_6, \ldots, s_i, \ldots \quad s_{49}, \quad s_{50}\}$$

So, the set $\{s_1, \quad s_2, \quad s_3, \quad s_4, \quad s_5\}$ is the amount of supply and demand for the network with 5 nodes with $s_1 = \sum_{i=2}^{5} s_i$.

- L_{ij} (The length between node i and node j): The distances between nodes in a network are represented as a symmetric matrix. We generate 10 different symmetric matrices L, with the dimension of 50×50. The elements of this matrix represent the distances between node i and node j (L_{ij}) that are generated randomly from uniform distribution of the interval of $[50, 3000]$.

$$\mathrm{L} = [\mathrm{L_{ij}}]_{50\times 50} = \begin{bmatrix} 0 & L_{1,2} & L_{1,3} & L_{1,4} & L_{1,5} & \ldots & L_{1,j} & \ldots & L_{1,50} \\ L_{2,1} & 0 & L_{2,3} & L_{2,4} & L_{2,5} & \ldots & L_{2,j} & \ldots & L_{2,50} \\ L_{3,1} & L_{3,2} & 0 & L_{3,4} & L_{3,5} & \ldots & L_{3,j} & \ldots & L_{3,50} \\ L_{4,1} & L_{4,2} & L_{4,3} & 0 & L_{4,5} & \ldots & L_{4,j} & \ldots & L_{4,50} \\ L_{5,1} & L_{5,2} & L_{5,3} & L_{5,4} & 0 & \ldots & L_{5,j} & \ldots & L_{5,50} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{i,1} & L_{i,2} & L_{i,3} & L_{i,4} & 0 & \ldots & L_{i,j} & \ldots & L_{i,50} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{50,1} & L_{50,2} & L_{50,3} & L_{50,4} & L_{50,5} & \ldots & L_{50,j} & \ldots & 0 \end{bmatrix}_{50\times 50}$$

Our test data is for $n = 5$, 10, 15, 20, 25, and 50. The distance's matrices for values of $n \leq 25$ is obtained from the $n = 50$ data by taking the elements of:

$$\mathrm{L} = [\mathrm{L_{ij}}]_{\mathrm{n}\times\mathrm{n}} = \begin{bmatrix} 0 & L_{1,2} & L_{1,3} & \ldots & L_{1,n} \\ L_{2,1} & 0 & L_{2,3} & \ldots & L_{2,n} \\ L_{3,1} & L_{3,2} & 0 & \ldots & L_{3,n} \\ \vdots & \vdots & \vdots & 0 & \vdots \\ L_{n,1} & L_{n,2} & L_{n,3} & \ldots & 0 \end{bmatrix}_{n\times n}$$

For example, for the distance matrix of 5 nodes, we select the first five

elements of the rows and columns of the matrix L with the dimension of 50×50 that is:

$$\mathrm{L} = [\mathrm{L_{ij}}]_{5\times5} = \begin{bmatrix} 0 & L_{1,2} & L_{1,3} & L_{1,4} & L_{1,5} \\ L_{2,1} & 0 & L_{2,3} & L_{,24} & L_{2,5} \\ L_{3,1} & L_{3,2} & 0 & L_{3,4} & L_{3,5} \\ L_{4,1} & L_{4,2} & L_{4,3} & 0 & L_{4,5} \\ L_{5,1} & L_{5,2} & L_{5,3} & L_{5,4} & 0 \end{bmatrix}_{5\times5}$$

- d, $C(d)$ (Pipe diameter type d and its per unit length dollar cost): These values are obtained from Mohajeri et al. (2012) that are 11 different pipe diameter types with their per unit length cost. Table 4.1 presents the pipe diameter type d and its per unit length cost.

.

Table 4.1: Pipe diameter type d and its per unit length cost

Pipe Diameter Type	Diameter(mm)	Per unit length (m) cost ($)
1	63	11.6
2	90	14.2
3	110	18.5
4	125	22
5	160	28
6	178	38
7	203	53
8	254	80
9	305	125
10	400	150
11	500	180

- π_i^L, π_i^U (Lower and Upper square pressure at node i): These values are 160000 and 360000 (Pa^2), respectively that are obtained from Wu et al. (2007).

- M (A large number): This is a value that is considered greater than the amount of supply node (s_1).

Consequently, for each size network, we generate 10 different symmetric matrices L as length matrices with three demands scenarios. We test our model for 30 instances, (10×3), of each size network.

Different Scenarios for the OA/ER/AP Algorithm to Test Our Mathematical Model

We build our MINLP mathematical model (Model 6) in AIMMS commercial software. AIMMS was introduced as a mathematical modelling tool in 1993. AIMMS has proven to be one of the world's most advanced development environments for building optimisation based decision support applications and advanced planning systems (Bisschop and Roelofs (2006)).

The OA/ER/AP algorithm is commercially available through AIMMS called AIMMS Outer Approximation (AOA) algorithm. The AOA algorithm is implemented using interplay between two solvers, the CPLEX solver for solving the Mixed Integer Linear programming and the CONOPT solver for solving the NLP programmings. The algorithm terminates if the objective of the Master MILP problem becomes larger than the objective of the NLP problem (in a case of minimisation) or the Master MILP problem is infeasible within the specified time limit.

In this algorithm, we consider the effective parameters in the NLP subproblems and the stopping criteria in the Master MILP subproblem. The different scenarios that are used to solve our MINLP model (Model 6) are outlined below.

- We consider Single start and Multi start modules for the nonlinear solver (CONOPT). As the NLP sub-problem has multiple local solutions, the algorithm is not guaranteed to find the global optimum. Practical experience has shown that it is sometimes difficult to get a feasible or better solution to the NLP sub-problems (Kan and Timmer (1987)). Therefore, if we consider a number of starting points for the NLP problem, then there may be a much better chance of finding a good feasible solution. More detailed definitions of starting points for the NLP problem has been given in Section (3.3.2).
- We set different values of the MILP Relative Optimality Tolerance ROT (MILP) as the stopping criteria tolerance value for the Master MILP solver (CPLEX). The iteration procedure for solving the MILP problem is to relax the integer variables and then solve the linear problem model. The relaxed LP of our Master MILP (M-Model 6) problem provides the lower bound

(RLB) for our Master MILP problem. The upper bound here is the objective function value of the previously solved the NLP sub-problem. We define the Relative Optimality Tolerance to guarantee that the value of RLB lies within a certain percentage of the value of UB. Sometimes the Master MILP problem finds a good integer solution early but must examine many additional nodes to prove the solution is optimal and this leads to taking more time without any improvement in the (RLB). We can speed up the process changing the optimality tolerance (Bisschop and Roelofs (2006)). In Section (3.6), a more detail description is given.

Table 4.2 indicates the size of the problem for each category.

Table 4.2: The test problem sizes for $n = 5$, 10, 15, 20, 25, and 50

n	Constraints	Variables	Non Zero Values
5	13	53	152
10	20	93	263
15	30	143	408
20	40	193	553
25	50	291	842
50	100	689	2011

The non zero values relate to the coefficients in the formulations of our model. Table 4.2 shows that increasing the number of consumers results in a significant increase in the size of the problem. Obviously, we need to consider specific parameters and stopping criteria for each sized test problem to find a good feasible solution.

4.2 The NLP-Single Starting Point

In this section, we present the computational results for solving our model with different test cases using the OA/ER/AP algorithm through AIMMS (AOA). We test our model with 30 instances for each network with 5, 10, 15, 20 (small network size), 25, and 50 nodes. Each network includes one source node and $n - 1$ demand nodes. In terms of the NLP sub-problems that has been discussed in Section (3.3.2), we consider the Single start module for testing our model with different network sizes. Also, we consider different optimality gap, the value of ROT(MILP), for each test network sizes to find a cost-effective feasible solution

in a reasonable computation time.

Relative Optimality Tolerance for the Master MILP sub-problem

In terms of setting the value of the MILP relative optimality tolerance ROT(MIP), we need to consider the size of the network. The computational results, here, are aimed to determine the value of ROT(MIP) for different test network sizes to find a cost-effective feasible solution. We clarify the purpose of our computational analysis below:

1. We test all instances for different sized test networks when the value of ROT(MILP) is the default value of 10^{-13}. The aim here is to find the maximum size test networks that the AOA algorithm can solve all instances in a reasonable time. We show that the AOA algorithm is able to solve all instances of test networks with $n \leq 20$ nodes with ROT(MILP)= 10^{-13} within 24 hours.

2. For $n > 20$ we consider networks with 25 and 50 nodes. We observe that the AOA algorithm cannot solve any test case of these sizes with ROT(MILP)=10^{-13} in a reasonable time. Therefore, the aim is to find the bounding values U and L for the ROT(MILP) for networks with 25 and 50 nodes that give:

   - The U value that is the smallest value so that all test instances are solved in less than 24 hours using the AOA algorithm. Obviously, all test instances are solved when the value of ROT(MILP) is greater than value U (ROT(MILP)) $> U$.

   - The L value that is the greatest value so that some test instances are solved in less than 24 hours. The AOA algorithm cannot solve any test instances when the value of ROT(MILP) is less than the value L (ROT(MILP) $< L$) in a reasonable time.

We explain the values of ROT(MILP) for networks with 25 and 50 nodes in more detail in Sections 4.2.2 and 4.2.3. Moreover, we show the Lower and Upper bounding values for ROT(MILP) based on tested instances for specific values of ROT(MILP) are as follows:

Table 4.3: Optimality gap values (25 and 50 nodes)

n	L	U
25	0.01	0.05
50	0.1	0.2

Applying the AOA method, the output results for each instance are the values for the variables z_{ij}, f_{ij}, π_i, and l_{ij}^d. In the next sections, we report the computational results of our test problem including the objective function values and their computation times.

In the next section, we present the computational results for test networks with 5, 10, 15, and 20 nodes (small sizes).

4.2.1 Small Size Networks

Table 4.4 presents the Minimum, Average, and Maximum of the computation times of all 30 instances for each network size. The computational results are obtained under the Single start condition and the default value of 10^{-13} for the ROT(MILP). Table 4.4 shows that the AOA algorithm works well for the small size test networks in determining the local optimum in a reasonable computation time. The AOA algorithm solves all 30 test cases with the default value ROT(MILP)= 10^{-13} in a reasonable time. In the next section, we provide the computational results for test networks with 25 nodes.

Table 4.4: The computation times (small size networks)

n	Time (sec)		
	Minimum	Average	Maximum
5	0.79	1.07	1.72
10	1.24	12.63	29.97
15	1.93	31.54	157.56
20	28.58	691.36	3028.58

4.2.2 25 Node Networks

For $n \geq 25$, the AOA algorithm cannot solve our model with ROT(MILP)=10^{-13} in a reasonable time. We test our model with 30 instances of networks with 25 nodes when the value of ROT(MILP) is 0.05. We demonstrate that all instances are solved by the AOA algorithm when the values of ROT(MILP) are greater than 0.05. Then, the model is tested for each instance with ROT(MILP)= 0.03 and 0.01. We select the instances that have a better chance to be solved (the ones that are solved in less computational time than other instances with ROT(MILP)= 0.03 and 0.01). We test these selected instances with ROT(MILP) less than 0.01 such as 0.009 and 0.008. The AOA does not solve any of these instances in a reasonable computational time. As shown in Table 4.5, we present the improvement of the objective function value (minimum cost) and of the computation time by percentage between the results with ROT(MILP) = {0.03 and 0.01} and with ROT(MILP) = {0.05}. The comparative analysis is based on:
Improvement=

$$\frac{\{\text{Results with ROT(MILP)=0.05}\}\text{-}\{\text{Results with ROT(MILP)= 0.03, 0.01}\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values of the results with ROT(MILP) = 0.05 and ROT(MILP) = 0.03, and 0.01. Also, the positive and negative values show improvement and deterioration respectively based on the results with ROT(MILP) = 0.05. In Table 4.5, we show the computational results considering the NLP-Single start for networks with 25 nodes when we consider different values for the ROT(MILP).

Table 4.5: Improvement/deterioration in the objective function value and the computation time (25 nodes)

	Improvement			
	ROT(MILP)=0.03		ROT(MILP)=0.01	
Instance	Min Cost%	Time (sec)%	Min Cost%	Time (sec)%
1	4.12	-304.25	4.12	-608.09
2	-14.89	6.03	-20.10	-18.86
3	0.28	-259.20	-17.68	-89.80
4	1.44	16.03	1.36	-85.10
5	0.00	-7.09	-12.89	-808.84
6	0.00	-376.39	-0.20	-935.81
7	0.00	-175.75	0.00	-799.07
8	0.00	-85.71	0.00	-92.50
9	-30.73	-324.02	-33.16	-602.29
10	0.00	-0.25	0.00	-138.42
11	0.00	-32.16	0.00	-524.58
12	0.00	-208.10	0.00	-681.88
13	-8.65	-36.78	-2.89	-92.68
14	-18.08	-967.23	5.69	-861.14
15	0.00	-215.08	73.16	-685.33
16	-7.90	-34.79	-7.47	-49.65
17	-2.35	59.37	-2.35	0.02
18	0.00	0.00	0.96	5.32
19	-1.99	26.94	-1.99	-0.70
20	0.00	19.01	0.00	-84.33
21	0.00	-98.19	0.00	-208.85
22	1.20	-75.68	22.72	-22.81
23	0.00	-59.09	0.00	-194.46
24	0.00	-273.59	0.00	-294.36
25	0.00	-1.04	0.00	-49.11
26	0.00	-10.65	0.00	15.54
27	34.68	-942.35	34.68	-1457.36
28	9.71	-8.83	8.06	-193.72
29	0.00	-43.83	0.00	-125.45
30	0.00	-152.86	0.00	-251.50

Table 4.5 indicates that for all test cases, the computation times increase when the value of ROT(MILP) is decreased. In terms of the objective function value, there are improvements in some test cases when the value of ROT(MILP) is decreased to 0.03 or 0.01.

We note that all 30 instances out of 30 test cases will be solved when the value of ROT(MILP) is greater than 0.05 (ROT(MILP) $>$ 0.05). Also, based on our experiments, we need more than 24 hours to find a feasible solution for all test problem instances when the value of ROT(MILP) is less than 0.01 (ROT(MILP) $<$ 0.01).

Moreover, Table 4.6 shows the number of instances out of 30 that there is no change, deterioration and improvement in the cost with ROT(MILP)=0.01 and 0.03.

Table 4.6: Number of instances without/with a change in the cost (25 nodes)

Different ROT(MILP)	No Change	Deterioration	Improvement
0.03	17	7	6
0.01	13	9	8

As Table 4.6 shows, about half of 30 instances are not changed in the objective function value with ROT(MILP) = 0.05 compared with the ROT(MILP) = 0.01 and 0.03. There is the only improvement in the cost for 6 and 8 out of the 30 instances with ROT(MILP) = 0.03 and 0.01, respectively. In contrast, there is no improvement in the cost for 7 and 9 out of 30 instances with ROT(MILP) = 0.03 and 0.01, respectively.

Table 4.7 shows the Minimum, Average, and Maximum of the computation times of all 30 instances when we consider different values for the ROT(MILP).

Table 4.7: The computation times (25 nodes)

Time (sec)	ROT(MILP)=0.05	ROT(MILP)=0.03	ROT(MILP)=0.01
Minimum	376.38	417.97	713.47
Average	3,459.74	7,551.38	11,324.43
Maximum	14,915.26	31,234.07	42,341.74

Table 4.7 shows as we are expecting the computation time increases when the value of ROT(MILP) is declines. The average computation time with the ROT(MILP) = 0.05 is about one hour and this increases to 2 and 3 hours with ROT(MILP) = 0.03 and 0.01, respectively.

In the next section, we present the computational results for networks with 50 nodes considering the NLP-Single start and different values of ROT(MILP).

4.2.3 50 Node Networks

This data size is considerable for our model as we have one supply node in our network. We first test our model for all 30 instances with ROT(MILP)=0.2. Then, we test the instances for the values of ROT(MILP) less than 0.2 again to generate a cost-effective feasible solution. We select the instances that are solved with ROT(MILP)= 0.1. We test these selected instances with ROT(MILP) less than 0.1 such as 0.09 and 0.08. The AOA does not solve any of these instances in a reasonable computational time. We detail below the performance of the AOA algorithm when we consider the different values of ROT(MILP).

- When ROT(MILP) = 0.2, then all 30 instances are solved.
- When ROT(MILP) = 0.15, then 17 out of 30 instances are solved.
- When ROT(MILP) = 0.1, then 11 out of 30 instances are solved.
- When ROT(MILP) $<$ 0.1, then none of the 30 instances are solved.

Note that we select the instances that have already been solved with ROT(MILP)= 0.1 and test them with ROT(MILP) less than 0.1 such as 0.09. The AOA does not solve these instances in a reasonable computational time.

Table 4.8 details the improvement by percentages in the objective function value and its computation time for all instances in the different values of ROT(MILP) compared to the results with ROT(MILP)=0.2. Moreover, 'NS' (No Solution) in Table 4.8 shows the instances that are not solved in a reasonable time (less than 24 hours). The comparative analysis is based on:

Improvement =

$$\frac{\{\text{Results with ROT(MILP)=0.2}\}\text{-}\{\text{Results with ROT(MILP)} = 0.15, 0.1\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values of the results with ROT(MILP) = 0.2 and with ROT(MILP) = 0.15, and 0.1. Also, the positive and negative value shows improvement and deterioration respectively based on the results with ROT(MILP) = 0.2. Table (4.8) shows the computational results with the NLP-Single start for networks with 50 nodes when we consider different values of ROT(MILP).

Table 4.8: Improvement/deterioration in the objective function value and the computation time for 50-nodes network. Note that 'NS' means No Solution obtained within the specified time limit.

	Improvement			
	ROT(MILP)=0.15		ROT(MILP)=0.1	
Instance	Minimum Cost%	Time (sec)%	Minimum Cost%	Time (sec)%
1	294.63	-43.33	90.11	-2750.65
2	-6.40	-27.24	-12.39	-2390.50
3	0.00	-128.63	NS	NS
4	-4.97	-35.25	-5.54	-1500.99
5	0.00	-78.96	3.19	-5682.42
6	0.00	-173.07	NS	NS
7	-5.36	-52.61	4.91	-2388.99
8	5.01	-129.05	NS	NS
9	-3.28	-522.54	NS	NS
10	0.00	-165.58	0.00	-863.17
11	0.00	-162.20	3.62	-5386.85
12	0.00	-87.98	0.00	-2420.67
13	-4.33	-238.93	-5.82	-8464.18
14	0.00	-577.09	NS	NS
15	0.00	-185.86	0.00	-7891.01
16	0.00	-249.51	0.00	-1278.80
17	0.00	-916.95	NS	NS
18	0.00	-2967.20	NS	NS
19	-4.43	-474.64	NS	NS
20	0.00	-1080.17	NS	NS
21	NS	NS	NS	NS
22	-5.44	-37.61	NS	NS
23	0.00	-845.98	NS	NS
24	NS	NS	NS	NS
25	0.00	-469.19	NS	NS
26	NS	NS	NS	NS
27	NS	NS	NS	NS
28	0.00	-1067.53	NS	NS
28	NS	NS	NS	NS
29	NS	NS	NS	NS
30	NS	NS	NS	NS

Table 4.8 indicates that, when the value of ROT(MILP) is decreased, some test cases are not solved within the 24 hours time limit. Based on tested instances for specific values of ROT(MILP), the bounding values U and L for the networks with 50 nodes are 0.2 and 0.1, respectively, such that:

- The value of ROT(MILP) = 0.2 is the smallest value for which all test instances are solved within 24 hours.
- The value ROT(MILP) = 0.01 is the greatest value for which at least one test instance is solved within 24 hours.

We note that all test instances will be solved when the value of ROT(MILP) is greater than 0.2 (ROT(MILP) $>$ 0.2). Also, when the value of ROT(MILP) is

less than 0.1 (ROT(MILP) < 0.01), no instance is solved within 24 hours.

Moreover, Table 4.9 shows the number of instances out of 30 such that there is no change, deterioration, and improvement in the objective function value with ROT(MILP)=0.15 and 0.1.

Table 4.9: Number of instances without/with a change in the cost (50 nodes)

Different ROT(MILP)	Not solved	No change	Deterioration	Improvement
ROT(MILP)=0.15	7	14	7	2
ROT(MILP)=0.1	19	4	3	4

Table 4.9 shows that decreasing the value of ROT(MILP) from 0.2 to 0.15 and then to 0.1 results in no change in the objective function value in the number of 14 and 4 test cases out of 30 instances. The only improvements in the total cost are for 2 and 4 out of the 30 instances when the values of ROT(MILP) are 0.15 and 0.1, respectively. In contrast, there are deterioration in the cost for 7 and 3 out of the 30 instances when the values of ROT(MILP) are 0.15 and 0.1, respectively.

Table 4.10 indicates the Minimum, Average, and Maximum of the computation times of all 30 instances when we consider different values of ROT(MILP).

Table 4.10: The computation times (50 nodes)

Time (sec)	ROT(MILP)=0.2	ROT(MILP)=0.15	ROT(MILP)=0.1
Minimum	669.32	905.27	7,295.22
Average	4,348.42	12,112.13	34,642.52
Maximum	40,362.54	69,233.20	73,991.10

Table 4.10 shows the computation time increases when the value of the ROT(MILP) declines. The average computation time among all solved test cases with the ROT(MILP) = 0.2 is about 1.2 hours and then increases to 3.3 and 9.6 hours with the ROT(MILP) = 0.15 and 0.1, respectively.

Conclusion

We present some concluding remarks in the computational results under the conditions of the single start for the NLP sub-problem and the different values of ROT(MILP). For the network with the size $n \leq 20$, the AOA algorithm solves all test instances in a reasonable computation time with the default value of 10^{-13}. When the value of ROT(MILP) is 10^{-13}, the AOA does not solve any network greater than 20 nodes within 24 hours. Therefore, we consider a larger value than the default value for ROT(MILP) to test all test instances with 25 and 50 nodes. For each size network, we examined all test cases with three different values of ROT(MILP) with the aim of generating a cost-effective feasible solution for each case.

Based on earlier provided details in Section 4.2.2 for the test networks with 25 nodes, the AOA algorithm needs more than 24 hours to find a feasible solution for our test cases when the value of ROT(MILP) is less than 0.01. In contrast, for the value of ROT(MILP) greater than 0.05, all test instances are solved.

Moreover, for the networks of 50 nodes, if the value of ROT(MILP) is greater than 0.2, then all test instances are solved within 24 hours. When the value of ROT(MILP) is 0.15 and 0.1, some instances are not solved within 24 hours.

Table 4.11 summarises the performance of the OA/ER/AP algorithm (AOA) using different values of ROT(MILP) for different test network sizes. We note that:

- For $10^{-13} \leqslant$ ROT(MILP), all test networks with less than and equal 20 nodes are solved.
- For $0.01 \leqslant$ ROT(MILP), all test networks with 25 nodes are solved.
- For $0.2 \leqslant$ ROT(MILP), all test networks with 50 nodes are solved.

Table 4.11: Computational results with the NLP-Single start and different values of ROT(MILP)

n	5	10	15	20	25			50		
ROT(MILP) value	10^{-13}	10^{-13}	10^{-13}	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of solved instances	30	30	30	30	30	30	30	11	17	30
Minimum time (sec)	0.79	1.24	1.93	28.58	713.47	417.97	376.38	7,295.22	905.27	669.32
Average time (sec)	1.07	12.63	31.54	691.36	11,324.43	7,551.38	3,459.74	34,642.52	12,112.13	4,348.42
Maximum time (sec)	1.72	29.97	157.56	3,028.58	42,341.74	31,234.07	14,915.26	73,991.10	69,233.20	40,362.54

4.3 The NLP-Multiple Starting Points

As explained in Section 3.3.2, we can use the Multi start option for our MINLP model to increase the chance of success in finding a cost effective feasible solution. Based on the Multi start algorithm (Algorithm 2) detailed in Chapter 3, the NLP problem is solved with different initial points in parallel and we choose the best solution from these points. The initial points are the best points in terms of penalised objective value selected from the generated sample points. The input data for this algorithm are the number of sample points (N_s) and the number of selected sample points (γN_s). The default value for N_s is 10 and γN_s is 5. We find pair values of N_s and γN_s to generate a cost-effective (best among the others) feasible solution for all sized test problems.

For the small size networks with at most 20 nodes, we consider the default value for the N_s and for the γN_s as 10 and 5, respectively. The default value of N_s and γN_s should be greater for the larger size problem (for n= 25 and 50). These values are determined as follows:

1. Consider different values of the N_s and fix the value of γN_s. In our work, we consider the values $N_s \in \{10, 15, 20, 25, 30, 50\}$ when $\gamma N_s \in \{5, 10, 15\}$ is fixed for $n = 25$. Also, we consider the values $N_s \in \{10, 20, 50, 100, 150\}$ when $\gamma N_s \in \{5, 10\}$ is fixed for $n = 50$.
2. Pick five instances from the uniform distribution of $[1, 30]$, among the 30 test cases for $n = 25$ and $n = 50$.
3. Test these five instances using different values of the N_s and γN_s with the NLP-Multi start method.
4. Determine the average minimum cost and the average computation time using the results of the given five instances. The reason we consider the average is that the minimum cost and computation time almost falls at the same value of N_s and γN_s for all five instances.

We will show further in this chapter that for a network with 25 nodes, the best comparison value for N_s is 20 and for γN_s is 15. Also for a network with 50 nodes, the best comparison value for N_s is 150 and for γN_s is 5. We test all 30 instances based on the values of N_s and γN_s that are determined based on the above steps.

4.3.1 Small Size Networks

Table 4.12 presents the Minimum, Average, and Maximum of computation time of all 30 instances for small sizes. All instances are tested when we consider the default value of $N_s = 10$ and $\gamma N_s = 5$ for the NLP-Multi start and the default value of 10^{-13} for ROT(MILP).

Table 4.12: The computation times (Small size networks)

	Time (sec)		
n	Minimum	Average	Maximum
5	0.24	0.57	1.33
10	1.97	13.89	40.17
15	5.04	258.82	2,047.00
20	52.02	12,316.96	62,745.77

The results in Table 4.12 show that the AOA algorithm works well for the small size networks in terms of finding a feasible solution for minimising the objective function (total cost) within 24 hours. In the next section, we present the computational results for the network with 25 nodes.

4.3.2 25 Node Networks

We first need to pick five instances from the uniform distribution of $[1, 30]$. Then we test the model for the different values of N_s and γN_s. We test these five instances with ROT(MILP) = 0.05 as we know all test instances are solved with this optimality gap value.

Table 4.13 shows the average results for five test instances of 25 nodes with different values of N_s and γN_s under situations of the NLP-Multi start and ROT(MILP) = 0.05. Table 4.13 indicates that with different number of sample points (N_s) and fixed selected sample points (γN_s), the cost increase with a significant decrease in the computation time. Also, Table 4.13 shows that the case of $N_s = 20$ and $\gamma N_s = 15$ gives the minimum objective function value. In this case the computation time is 1137.84 (sec). Therefore, we test all 30 instances of networks with 25 nodes with $N_s = 20$ and $\gamma N_s = 15$.

Table 4.13: Multiple starting points for 25 node networks

γN_s						
	5		10		15	
N_s	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)
10	87611.23	2400.60	89482.35	1027.58	-	-
15	96199.61	1600.33	94148.88	1824.37	97841.42	1963.41
20	95585.17	1404.36	92088.42	1014.68	87531.58	1137.84
30	89030.93	2579.15	93753.09	1090.30	87668.29	1633.39
50	93208.27	1978.72	90528.02	2739.69	93883.44	1547.45

We note that we test our instances when $\gamma N_s \leqslant N_s$. In Table 4.13, we have the situation that $N_s \leqslant \gamma N_s$ $(10 < 15)$ and we cannot test our five instances. In Table 4.14, we show the computational results for the network with 25 nodes with the NLP-Multi start and different values of ROT(MILP).

Table 4.14 details the improvement by percentages in the minimum cost and its computation time for all instances with different values of ROT(MILP) compared to the results with ROT(MILP)=0.05. Moreover, 'NS' (No Solution) in Table 4.14 shows the instances that are not solved within the 24 hours time limit. The comparative analysis is based on:
Improvement =

$$\frac{\{\text{Results with ROT(MILP)=0.05}\}\text{-}\{\text{Results with ROT(MILP)= 0.03, 0.01}\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values of the results with ROT(MILP) = 0.05 and with ROT(MILP) = 0.03, and 0.01. Also, the positive and negative value shows improvement and deterioration value in both cost and computation time, respectively based on the results with ROT(MILP) = 0.05.

Table 4.14: Improvement/deterioration in the objective function value and the computation time for 25-nodes networks. Note that 'NS' means No Solution obtained within the specified time limit.

	Improvement			
	ROT(MILP)=0.03		ROT(MILP)=0.01	
Instance	Minimum Cost%	Time(sec)%	Minimum Cost%	Time(sec)%
1	10.71	-118.44	5.68	-188.88
2	-12.23	-709.22	-20.90	-207.38
3	-7.14	-278.88	-20.39	-77.54
4	-1.51	-79.55	-1.47	-113.50
5	0.00	-24.45	-8.57	-786.01
6	-17.18	-11.20	-17.18	-142.76
7	-0.12	-138.09	-0.12	-554.42
8	0.00	-32.90	0.00	-77.39
9	0.91	-99.01	NS	NS
10	0.00	-223.08	0.00	-135.90
11	0.00	-89.51	0.00	-296.96
12	0.00	-69.99	0.00	-120.04
13	16.73	-85.74	11.66	-327.54
14	-0.00	-293.73	36.79	-302.04
15	-2.82	31.37	-2.82	-136.50
16	2.77	9.48	13.48	-48.23
17	-4.56	-26.15	-1.95	-35.99
18	0.00	25.93	0.96	-135.99
19	-1.16	-2305.60	3.21	-2386.47
20	0.00	-83.05	0.00	-153.64
21	0.00	23.60	0.00	-1.20
22	0.00	-36.04	0.44	-441.79
23	0.00	-116.56	0.00	-263.66
24	0.00	-45.27	0.00	-513.37
25	0.00	69.15	0.00	-67.25
26	0.00	-239.97	0.00	-301.92
27	0.00	19.83	0.00	-46.15
28	12.49	-111.37	12.49	-231.33
29	0.00	-374.12	0.00	-263.88
30	0.00	123.55	0.00	-126.75

Tables 4.14 shows when the value of ROT(MILP) is decreased from 0.05 to 0.03 and then to 0.01, the computation time increases. In addition, all 30 instances are solved when ROT(MILP)$\geq$ 0.03. Among all test cases, there is only one case that is not solved within 24 hours (instance 9) with ROT(MILP) = 0.01.

Table 4.15: Number of instances without/with a change in the cost

Different ROT(MILP)	Not Solved	No change	Deterioration	Improvement
ROT(MILP)=0.03	0	18	8	4
ROT(MILP)=0.01	1	13	8	8

As Table 4.15 shows when the value of ROT(MILP) is decreased from 0.05 to 0.03 and then 0.01, about half of the instances of 30 are not changed in the cost. Also, 4 instances are improved in the cost when ROT(MILP) = 0.03. Although, this number is increased to 8 when the ROT(MILP) is 0.01 or 0.03. Table 4.16 presents the Minimum, Average, and Maximum of computation times.

Table 4.16: The computation times (25 nodes)

Time (sec)	ROT(MILP)=0.05	ROT(MILP)=0.03	ROT(MILP)=0.01
Minimum	46.41	526.67	528.91
Average	3,459.74	7,551.38	11,813.09
Maximum	13,252.97	23,697.07	72,924.10

Table 4.16 shows when the value of ROT(MILP) is decreased from 0.05 to 0.03 and then 0.01, then the average computation time increases from one hour to 2 and 3.2 hours, respectively. In the next section, we present the computational results for a network with 50 nodes.

4.3.3 50 Node Networks

We first need to pick five instances from the uniform distribution of $[1, 30]$. Then, using these selected instances, we test our model with different values of N_s and γN_s. We test all test instances with ROT(MILP)=0.2 as we know all test instances are solved in this optimality gap value.

Table 4.17 shows the average results for five test instances of 50 nodes with different values of N_s and γN_s under situations of the NLP-Multi start and ROT(MILP)=0.2. Also, Table 4.17 shows, the case of $N_s = 150$ and $\gamma N_s = 5$ gives minimum objective function value. In this case, the computation time is $1,225.928$. Therefore, we test all 30 instances of networks with 50 nodes when $N_s = 150$ and $\gamma N_s = 5$.

Table 4.17: Multiple starting points for 50 node networks

	γN_s			
	5		10	
N_s	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)
10	210,820.96	1,983.97	212,534.09	2,221.73
20	217,081.98	1,266.60	215,414.42	1,764.75
50	213,805.92	1,151.84	214,909.89	1,743.59
100	216,555.13	1,194.58	215,544.47	1,681.40
150	206,013.19	1,225.92	210,075.33	2,312.39

Table 4.18 details the improvement by percentages in the minimum cost and its computation time for all instances with different values of ROT(MILP) compared to the results with ROT(MILP)=0.2. Moreover, 'NS' (No Solution) in Table 4.18 shows the instances that are not solved within the 24 hours time limit. The comparative analysis is based on:
Improvement =

$$\frac{\{\text{Results with ROT(MILP)=0.2}\}\text{- }\{\text{Results with ROT(MILP)= } 0.15, 0.1\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values of the results with ROT(MILP) = 0.2 and with ROT(MILP) = 0.15, and 0.1. Also, the positive and negative value shows improvement and deterioration value in both cost and computation time, respectively based on the results with ROT(MILP) = 0.2.

Table 4.18: Improvement/deterioration in the objective function value and the computation time for 50-node networks. Note that 'NS' means No Solution obtained within the specified time limited.

	Improvement			
n	ROT(MILP)=0.15		ROT(MILP)=0.1	
Instance	Min Cost%	Time(sec)%	Min Cost%	Time(sec)%
1	-12.50	-25.90	-12.50	-1781.60
2	-4.66	-110.55	9.29	-3613.06
3	0.00	-37.85	0.00	-2153.15
4	0.26	-39.81	-16.34	-638.18
5	0.00	-72.89	1.60	-3132.06
6	-1.93	-134.96	NS	NS
7	-0.52	-104.13	-3.75	-268.72
8	-1.42	-280.79	NS	NS
9	6.03	-88.75	NS	NS
10	-7.12	-79.30	-7.12	-88.79
11	0.00	-45.41	0.00	-36026.41
12	0.00	-91.65	0.00	-1302.70
13	2.91	-26.26	-11.41	-3644.14
14	-7.57	-472.72	NS	NS
15	0.00	-118.85	NS	NS
16	0.00	-46.62	0.00	-1763.44
17	NS	NS	NS	NS
18	NS	NS	NS	NS
19	0.00	-70.20	NS	NS
20	NS	NS	NS	NS
21	NS	NS	NS	NS
22	52.66	-61.90	NS	NS
23	NS	NS	NS	NS
24	NS	NS	NS	NS
25	NS	NS	NS	NS
26	NS	NS	NS	NS
27	NS	NS	NS	NS
28	NS	NS	NS	NS
29	NS	NS	NS	NS
30	NS	NS	NS	NS

We note from Table 4.18 that as the value of ROT(MILP) is decreased from 0.2 to 0.15 and then to 0.1, the computation time increases. In addition, all instances are solved with ROT(MILP) = 0.2 and some instances are not solved within the 24 hours time limit with ROT(MILP) = 0.015 and 0.01. Moreover, Table 4.19 shows the number of instances (out of 30) without/with a change in the cost for network with 50 nodes with different values of ROT(MILP).

Table 4.19: Number of instances without/with a change in the cost (50 nodes)

Different ROT(MILP)	Not Solved	No change	Deterioration	Improvement
0.15	12	7	7	4
0.1	19	4	5	2

Table 4.19 illustrates that decreasing the value of ROT(MILP) results in decreasing the number of solved instances (out of 30) by the AOA algorithm. We also show that only a few number of instances are improved in the objective function value (4 and 2 respectively with ROT(MILP) = 0.15 and 0.1). The Minimum, Average and Maximum computation time among all solved test instances with different values of ROT(MILP) are presented in the table below.

Table 4.20: The computation times (50 nodes)

Time (sec)	ROT(MILP)=0.2	ROT(MILP)=0.15	ROT(MILP)=0.1
Minimum	1,122.83	1,449.11	4,243.66
Average	4,575.70	3,942.46	29,733.58
Maximum	34,349.24	16,617.43	51,112.12

We present the following conclusion results for all sized test networks with the NLP-Multi start and different values of ROT(MILP) .

Conclusion

We present some concluding remarks in the computational result under the multi start condition for the NLP sub-problem and the different values of ROT(MILP).

For the network with $n \leq 20$ nodes, the AOA algorithm solves all test instances in a reasonable computation time with the default value of 10^{-13}. When the value of ROT(MILP) is 10^{-13}, the AOA does not solve any network with $n > 20$ ($n = 25$ and 50) within 24 hours. So, we consider a larger value than the default value for ROT(MILP) to test all test instances with 25 and 50 nodes. For each network, we tested all test cases with three different values of ROT(MILP) with the aim of generating a cost-effective feasible solution for each case.

Based on earlier provided details in Section 4.3.2, for the test networks with

25 nodes, the AOA algorithm does not find a feasible solution for our test cases when the value of ROT(MILP) is less than 0.01 within 24 hours. In contrast, for the value of ROT(MILP) greater than 0.03, all test instances are solved. Moreover, for the networks of 50 nodes, if the value of ROT(MILP) is greater than 0.2, then all test instances are solved within 24 hours. Also, with the value of ROT(MILP) = 0.15 and 0.1, some instances are not solved within 24 hours.

Table 4.21 summarises the performance of the AOA algorithm with different values of ROT(MILP) for different test network sizes. We note that:

- For $10^{-13} \leqslant$ ROT(MILP), all test networks with less than and equal 20 nodes are solved.
- For $0.015 \leqslant$ ROT(MILP), all test networks with 25 nodes are solved.
- For $0.2 \leqslant$ ROT(MILP), all test networks with 50 nodes are solved.

4.4 Comparative Analysis

In this section, we compare the results between the NLP-Single start and the NLP-Multi start. The comparative analysis is based on the improvement of the cost and time using:

$$\text{Improvement} = \frac{\{\text{The NLP-Single Start}\} - \{\text{The NLP-Multi Start}\}}{\text{Best value}} \times 100\% \tag{4.1}$$

The best value is the minimum value of the results between the NLP-Single start and the NLP-Multi start. Also, the positive and negative value shows improvement and deterioration value in both cost and computation time, respectively based on the results with the NLP-Single start.

Table 4.21: The computational results with the NLP-Multi start method and different values of ROT(MILP) (small-sized networks)

n	5	10	15	20	25			50		
ROT(MILP) value	10^{-13}	10^{-13}	10^{-13}	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of solved instances	30	30	30	30	29	30	30	11	18	30
Minimum time (sec)	0.24	1.97	5.04	52.02	528.91	526.67	46.41	4243.66	1,449.11	1,122.83
Average time (sec)	0.57	13.89	258.82	12,316.96	11,813.09	6,674.81	3,966.85	29,733.58	3942.46	4,575.70
Maximum time (sec)	1.33	40.17	2,047.00	62,745.77	72,924.10	23,697.07	13,252.97	51,112.12	16,617.43	34,349.24

4.4.1 Small Size Networks

Table 4.22 shows the comparison in terms of computation time between the NLP-Single start and the NLP-Multi start for the small-sized test problem with ROT(MILP) = 10^{-13}.

Table 4.22: The comparison of the computation time between the NLP-Single start and the NLP-Multi start for $n \leq 20$.

n	5	10	15	20
Minimum time (sec)%	229.17	-58.87	-161.14	-82.02
Average time (sec)%	87.72	-9.98	-720.61	-1681.56
Maximum time (sec)%	29.32	-34.03	-1,199.19	-1971.79

From Table 4.22, it turns out that the NLP-Multi start gives an improved computation time value for the networks with 5 nodes. We note that the average improved time is 87.72%. Also, the NLP-Multi start gives changes in the average computation time of -9.98%, -720.61% and -1681.56% for the sized network with 10, 15, and 20 nodes, respectively. The computation time increases with the NLP-Multi start compared to the NLP-Single start.

4.4.2 25 Node Networks

In this part, we compare the results of objective function value and computation time from Sections 4.2 and 4.3 using the equation (4.1) to provide the improvement in the cost and computation time for the NLP-Multi start.

Table 4.23 details the improvement in the cost and computation time for the NLP-Multi start compared with the NLP-Single start for all 30 instances with 25 nodes considering the different values of ROT(MILP). Table 4.23 shows using the NLP-Multi start, 14, 14, and 13 test cases are improved in the time with ROT(MILP) = 0.05, 0.03, and 0.01, respectively. Also, 4, 4, and 5 test cases are improved in the cost and computation time with ROT(MILP) = 0.05, 0.03, and 0.01, respectively.

Table 4.23: The comparison results between the NLP-Single start and the NLP-Multi start for 25-node networks. Note that 'NS' means No Solution obtained within the specified time limit.

	Improvement					
	ROT(MILP)=0.05		ROT(MILP)=0.03		ROT(MILP)=0.01	
Instance	Min Cost%	Time(sec)%	Min Cost%	Time(sec)%	Min Cost%	Time(sec)%
1	-1.49	-150.54	4.76	-35.38	0.00	-2.21
2	1.08	316.38	3.49	-106.81	0.41	61.01
3	10.88	-70.29	3.20	-79.62	8.39	29.35
4	2.85	113.72	-0.13	2.59	0.00	85.30
5	0.00	7.62	0.00	-7.99	3.98	10.40
6	7.80	-73.12	-8.70	147.45	-8.48	146.46
7	0.12	-62.95	0.00	-40.70	0.00	-18.61
8	0.00	-5.77	0.00	31.67	0.00	2.26
9	-28.27	51.65	2.85	223.12	NS	NS
10	0.00	-21.62	0.00	-291.93	0.00	-20.34
11	0.00	-16.43	0.00	-66.95	0.00	35.13
12	0.00	-255.34	0.00	-96.06	0.00	0.00
13	-4.64	54.07	21.21	13.46	9.80	-44.02
14	-29.48	-187.06	-9.65	-5.90	-0.04	-20.07
15	2.82	-173.30	0.00	67.99	-73.16	21.50
16	-1.08	7.14	9.70	59.54	20.64	8.17
17	2.74	83.48	0.56	-17.62	3.14	34.89
18	0.00	14.46	0.00	44.14	0.00	-117.14
19	-0.82	2248.59	0.00	-30.02	4.41	-5.13
20	0.00	-18.76	0.00	-158.72	0.00	-63.41
21	0.00	3.45	0.00	153.40	0.00	215.71
22	22.72	211.72	21.26	302.54	0.44	-41.53
23	0.00	37.11	0.00	0.72	0.00	11.02
24	0.00	-42.20	0.00	80.85	0.00	-121.18
25	0.00	-85.97	0.00	-8.81	0.00	-108.59
26	0.00	131.99	0.00	-32.45	0.00	-100.17
27	34.68	-573.38	0.00	85.49	0.00	58.24
28	-4.10	1.95	-1.52	-90.51	0.00	-10.65
29	0.00	-21.98	0.00	-302.10	0.00	-96.88
30	0.00	-106.12	0.00	174.26	0.00	-32.96

Table 4.24 shows the number of instances out of 30, such that, there are Not Solved, No change, Deterioration, Improvement in the objective function value based on the NLP-Single start compared with the NLP-Multi start. The results in Table 4.24 are obtained from Table 4.23.

Table 4.24: Number of instances without/with a change in the cost (25 nodes)

Different ROT(MILP)	Not Solved	No change	Deterioration	Improvement
0.05	0	14	7	8
0.03	0	18	4	8
0.01	1	18	3	8

The results from this table demonstrate that the cost is changed for about the half of 30 instances compared with the NLP-Single start results. We also note that in Table 4.23, mostly, if an instance is not changed in the cost with ROT(MILP)=0.05, they are not changed in the cost too when the value of ROT(MILP) is 0.03 or 0.01. However, there are 8 out of 30 instances that are improved in the cost when we solve our model using the NLP-Multi start. In addition, all 30 instances are solved using the NLP-Single start for different values of ROT(MILP). In contrast, the NLP-Multi start does not solve one instance out of 30 with ROT(MILP) = 0.01 within 24 hours.

4.4.3 50 Node Networks

In this part, we compare the results of cost and computation time from Sections 4.2.3 and 4.3.3 using the equation (4.1). We define 'NS' for the test cases that are not solved within 24 hours either for the NLP-Single start or the NLP-Multi start.

Table 4.25 details the improvement in the cost and computation time with the NLP-Multi start compared with the NLP-Single start for all 30 instances of 50 nodes with different values of ROT(MILP). Table 4.25 shows that using NLP-Multi start, 9, 5, and 6 test cases (among the cases that are solved) are improved in the time with ROT(MILP) = 0.2, 0.15, and 0.1, respectively. Also, 6, 1, and 1 test cases are improved in the cost and computation time with ROT(MILP) = 0.2, 0.15, and 0.1, respectively.

Table 4.25: The comparison results between the NLP-Single start and the NLP-Multi start for 50-node networks. Note that 'NS' means No Solution obtained within the specified time limit.

	Improvement					
N	ROT(MILP)=0.2		ROT(MILP)=0.15		ROT(MILP)=0.1	
Instance	Min Cost%	Time(sec)%	Min Cost%	Time(sec)%	Min Cost%	Time(sec)%
1	309.84	-50.31	-8.33	-32.03	91.62	0.80
2	-3.48	-20.18	-98.85	-1.79	18.69	-79.17
3	0.00	-102.02	0.00	-21.81	NS	NS
4	10.23	-73.79	16.00	-79.65	0.00	24.79
5	6.56	-27.20	6.56	-22.89	4.91	40.66
6	1.93	5.06	0.00	22.10	NS	NS
7	0.00	-21.38	4.82	-62.36	-8.84	456.13
8	10.23	-19.80	3.49	-99.16	NS	NS
9	2.68	19.28	12.44	-19.75	NS	NS
10	7.12	-48.24	0.00	-0.09	0.00	-37.24
11	0.00	-39.10	0.00	22.86	-3.62	-815.85
12	0.00	-50.49	0.00	-53.43	0.00	19.41
13	-2.04	-49.03	5.22	80.12	-7.43	34.84
14	7.07	-56.90	-0.47	-32.71	NS	NS
15	0.00	-43.64	0.00	-9.97	NS	NS
16	0.00	-102.78	0.00	17.56	0.00	-174.05
17	0.00	-37.68	NS	NS	NS	NS
18	0.00	-1421.76	NS	NS	NS	NS
19	-4.43	4.16	0.00	78.37	NS	NS
20	0.00	-42.22	NS	NS	NS	NS
21	0.00	143.11	NS	NS	NS	NS
22	-54.36	-22.42	4.60	-44.03	NS	NS
23	3.77	2.85	NS	NS	NS	NS
24	0.00	4.36	NS	NS	NS	NS
25	-8.22	-99.07	NS	NS	NS	NS
26	0.00	-34.83	NS	NS	NS	NS
27	0.00	-1.70	NS	NS	NS	NS
28	17.15	21.28	NS	NS	NS	NS
29	12.66	21.41	NS	NS	NS	NS
30	2.57	427.31	NS	NS	NS	NS

Table 4.26 shows the number of instances out of 30 that are Not Solved, there is No change, there is Deterioration, and there is Improvement in the objective function value based on the NLP-Single start compared with the NLP-Multi start. The results from this table demonstrate that about the half of 30 instances are not changed in the cost in the NLP-Multi start results compared with the NLP-Single start results when ROT(MILP) = 0.2. However, the test cases without change are decreased when the value of ROT(MILP) is decreased. Moreover, using the NLP-Multi start, the 12 out of 30 instances are improved in the cost when

ROT(MILP) = 0.2 . Also, the cost improvement for test cases are decreased to 7 and 3 with ROT(MILP) = 0.15 and 0.1, respectively.

Table 4.26: Number of instances without/with a change in the cost (50 nodes)

Different ROT(MILP)	Not Solved	No change	Deterioration	Improvement
0.2	0	13	5	12
0.15	5	8	3	7
0.1	0	4	3	3

Table 4.26 also demonstrates that the NLP-single start has a better performance for only a few numbers of 30 test cases (column of deterioration). In addition, the number of not solved instances with the NLP-Multi start when ROT(MILP) = 0.15 increase to 5 test cases. Based on Tables 4.8 and 4.18, we note that if an instance is not solved using the NLP-Single start, the NLP-Multi start also would not solve it within 24 hours. We present the following conclusion of comparative analysis for all sized test networks between the NLP-Single start and the NLP-Multi start with different values of ROT(MILP).

Conclusion

In Sections 4.4.2 and 4.4.3, we showed the comparative analysis in the cost and computation time among all solved test cases out 30, between the NLP-Single start and the NLP-Multi start with different values of ROT(MILP). The comparison Tables 4.24 for 25 nodes showed 8 test cases are improved in the cost with different values of MIT(ROT). For a network with 50 nodes, Table 4.26 shows the NLP-Multi start gives an improved cost in 12 out of 30, 7 out of 18 and 3 out of 10 solved test cases with ROT(MILP) = 0.2, 0.15 and 0.1, respectively.

4.5 Conclusion

In this chapter, we aimed to test our MINLP model (Model 6) given by equations (3.12)-(3.20) using an approximation method. The algorithm that we used is Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method developed by Viswanathan and Grossmann (1990) that alternates between solving a MILP subproblem and one or two NLP subproblems. This algorithm can handle the nonconvexity of the problem at the level

of Master MILP subproblem. To demonstrate the feasibility and effectiveness of the solution for our mathematical model, we consider the multi starting points in the level of the NLP sub-problems and different values of the MILP Relative Optimality Tolerance ROT(MILP) as the stopping criteria tolerance value for the Master MILP problem within the specified time limit.

The starting point for the NLP problem has an important role in providing a cost-effective feasible solution. A single start implementation might be trapped in a local optimum, but the multiple calls provide a better chance of finding a good feasible solution. Computational results aim to validate the model and to find a cost-effective feasible solution in a reasonable computation time (within 24 hours) for all test cases. In this section, the computational results were undertaken considering different values of ROT(MILP) for the Master MILP sub-problem and under the conditions of Single start and Multi start for the NLP sub-problems.

We detailed how we generated the test data for our model. We provided computational results using the OA/ER/AP method. It turns out that the smaller sized networks are solved when the optimality gap is 10^{-13}. We consider a greater value than 10^{-13} for the optimality gap (the value of ROT(MILP)) for networks with 25 and 50 nodes. We test all 30 instances for specific values of ROT(MILP) under the NLP-Single start and the NLP-Multi start conditions to find a cost-effective feasible solution within 24 hours. As expected the NLP-Multi start yields an improved objective function value but requires more computational time than the NLP-Single start. Our main outcome is an effective computational model for small networks.

The following tables show the number of instances without/with a change in the cost and computation time for networks with 25 and 50 nodes based on the values of ROT(MILP) = 0.05 and 0.2, respectively when we consider the NLP-Single start and the NLP-Multi start. The following results have been extracted from Tables 4.5 and 4.14 for 25 nodes and Tables 4.8 and 4.18 for 50 nodes. Tables 4.27 and 4.28 show that under the NLP-Single start and the NLP-Multi start conditions, decreasing the value of ROT(MILP) results in an increase in the computation time.

Table 4.27: Number of instances without/with a change in the cost and computation time (25 nodes)

	The NLP-Single start		The NLP-Multi start	
Number of instances that has been :	ROT(MILP)=0.03	ROT(MILP)=0.01	ROT(MILP)=0.03	ROT(MILP)=0.01
Solved	30	30	30	29
Improved in the computation time	5	0	7	0
Improved in the cost	6	8	4	8
Deteriorated in the cost	7	9	8	8
Not changed in the cost	17	13	18	13

Table 4.28: Number of instances without/with a change in the cost and computation time (50 nodes)

	The NLP-Single start		The NLP-Multi start	
Number of instances that has been :	ROT(MILP)=0.15	ROT(MILP)=0.1	ROT(MILP)=0.15	ROT(MILP)=0.1
Solved	23	11	18	11
Improved in the computation time	0	0	0	0
Improved in the cost	2	4	4	2
Deteriorated in the cost	7	3	7	5
Not changed in the cost	7	3	7	5

The following tables show the number of instances with/without change in the cost and computation time based on the NLP-Single start compared with the NLP-Multi start with different values of ROT(MILP) for networks of 25 and 50 nodes.

Table 4.29: Number of instances without/with a change in the cost and computation time (25 nodes)

Number of instances that has been :	ROT(MILP)=0.05	ROT(MILP)=0.03	ROT(MILP)=0.01
Solved	30	30	29
Improved in the computation time	14	14	13
Improved in the cost	8	8	8
Deteriorated in the cost	7	4	3
Not changed in the cost	14	18	18

Table 4.30: Number of instances without/with a change in the cost and computation time (50 nodes)

Number of instances that has been :	ROT(MILP)=0.2	ROT(MILP)=0.15	ROT(MILP)=0.1
Solved	30	18	11
Improved in the computation time	9	5	6
Improved in the cost	12	7	3
Deteriorated in the cost	5	3	3
Not changed in the cost	13	8	4

Chapter 5

The New Heuristic Algorithm

In this chapter, we develop a new heuristic algorithm to solve our Mixed Integer Non Linear Programming (Model 6) given by equations (3.12)-(3.20). Our fundamental problem is, given a set of nodes and customer requirements, determine the optimal network. This includes the network layout, the multi diameter type for the connected link, the pressure at each node, and the flow through each link. The objective is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which in our case is a tree.

Our solution strategy is to reduce the level of difficulty by converting the MINLP problem to the Linear Programming (LP) problem through a bi-level heuristic algorithm in which the integer variables are generated in the outer level, and the remaining LP sub-problem is solved in the inner level.

Our proposed algorithm includes two levels. At the outer level, a tree network is generated, and at the inner level, we determine the diameters for the given links. Also we determine the pressure at each node and the flow through each selected link by solving the remaining linear problem.

This chapter is organised as follows: In Section 5.1, we present the problem features followed by our mathematical model (Model 6). Section 5.2 explains the solution strategy for the developed model. The detailed algorithm along with the implementation is presented in Sections 5.3 and 5.4. We test our methodology on a series of numerical examples. In Sections 5.5 and 5.6, the improvement of

our heuristic algorithm compared with the NLP-Single start and the NLP-Multi start are presented.

5.1 Model

We start this section by recalling our mathematical model (Model 6) detailed in Chapter 3. We first recall the following features of the problem, which were listed in Chapter 3.

- The pipe diameter configuration must follow a descending order.
- There are no compressor and no nozzles in the network.
- The network is in a steady state, the flow may vary from point to point, but do not vary with time.
- All pipes in a network are of the same type of material, such as all pipes are made of steel or ductile iron.
- Without loss of generality, the sum of demand of consumer nodes and the source node is zero.

The notation and terminology is as follows:
Sets:

$N = \{1\} \cup N_c$: The set of supply and demand nodes.

$N = \{1, 2, 3, \ldots, i, \ldots, n-1, n\}$.

N_c: The set of demand nodes $\{2, 3, \ldots, i, \ldots, n-1, n\}$;

obviously, $|N| = n = |N_c| + 1$.

D : The set of pipe diameter types $\{d_1, d_2, \ldots, d_{|D|}\}$.

Parameters:

s_1 : The amount of supply at source node 1.

s_i : The demand at consumer node i.

$C(d)$: The cost per unit length of pipe diameter type d.

L_{ij} : The length between node i and node j.

${\pi_i}^L$, ${\pi_i}^U$: The lower and upper square pressure limits at node i.

M : A large number.

Variables:

π_i : The square of gas pressure at node i.

f_{ij} : The flow rate from node i to node j.

z_{ij} : The binary variable indicating whether or not node i is serving node j;

$$z_{ij} = \begin{cases} 1, & \text{if } i \text{ is serving (linked to) } j \text{ (i} \rightarrow \text{j)}. \\ 0, & \text{otherwise.} \end{cases}$$

l_{ij}^d : The length of pipe diameter type d from node i to node j.

We need to note that if $l_{ij}^d = 0$, then there is not the pipe diameter type d for link (i, j). We recall the MINLP mathematical problem (Model 6) given by equations (3.12)-(3.20) in below:

Model 6

$$\text{Minimise} \quad \sum_{i\in N}\sum_{j\in N}\sum_{d\in D} C(d) l_{ij}^d \tag{5.1}$$

$$\text{Subject to:} \quad \sum_{j\in N} f_{ij} - \sum_{j'\in N} f_{j'i} = s_i, \qquad \forall i \in N \tag{5.2}$$

$$f_{ij} \leq M z_{ij}, \qquad \forall j \in N, \forall j \in N \tag{5.3}$$

$$\sum_{i\in N} z_{ij} = 1, \qquad \forall j \in N_c \tag{5.4}$$

$$z_{ij}(\pi_i - \pi_j) = \beta \sum_{d\in D} l_{ij}^d d^{-5} f_{ij}^2, \qquad \forall i \in N, \forall j \in N \tag{5.5}$$

$$\sum_{d\in D} l_{ij}^d = z_{ij} L_{ij}, \qquad \forall i \in N, \forall j \in N \tag{5.6}$$

$$\pi_i{}^L < \pi_i < \pi_i{}^U, \qquad \forall i \in N \tag{5.7}$$

$$f_{ij} \geq 0, \qquad \forall i \in N, \forall j \in N, \forall d \in D \tag{5.8}$$

$$l_{ij}^d \geq 0, \qquad \forall i \in N, \forall j \in N, \forall d \in D \tag{5.9}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in N, \forall j \in N. \tag{5.10}$$

We note that β is a constant number that was explained in Chapter 2, Equation (2.6).

5.2 Solution Strategy

The Model 6 is a MINLP model which is difficult to solve with exact or even approximate methods. It is, first of all, an integer problem due to the binary choice of opening the arcs. Secondly, this problem is nonlinear because of the nonconvex and nonlinear constraint (5.5). We develop a heuristic algorithm based on a Cross-Entropy algorithm as a solution approach. Our solution strategy has two stages:

- **The outer level:** In this level, we define a procedure to generate the link connection variables z_{ij} (Network Design Problem). It leads to presenting a matrix of binary values of a_{ij} (adjacency matrix $A = [a_{ij}]$). We detail this further in Section 5.3.1.
- **The inner level:** In this level, the MINLP model (Model 6) is converted into a model where the binary variables are fixed (connection links). The binary variables are obtained from the outer level. Then, this converted model that is the Allocation Problem is solved. The output results are the flows, the pressures, and the length of pipeline diameter types. We present more detail below.

The network structure is a tree; then the inner model is an LP problem. The flow variables, f_{ij}, are obtained through a system of equations (Equations 5.2, 5.3, and 5.8). In the outer level of our solution strategy, we generate tree networks (adjacency matrix A) based on a probability matrix. In the inner level, we use the LP solver in AIMMS to solve the inner minimisation that the results are f_{ij}, π_i, and l_{ij}^d. The inner level is implemented in AIMMS that includes a procedure and a linear mathematical model.

The procedure: Using the adjacency matrix A obtained in the outer level, the variables f_{ij} are determined separately from the system of linear equations including the constraints (5.2), (5.3), and (5.8).
Procedure 1: Solve the system:

$$\begin{cases} \sum_{(i,j)\in A} f_{ij} - \sum_{(j',i)\in A} f_{j'i} = s_i, & \forall i \in N \\ f_{ij} \leq M a_{ij}, \quad \forall (i,j) \in A \\ f_{ij} \geq 0, \quad \forall (i,j) \in A. \end{cases}$$

We solve the above system of linear equations to obtain the flow (f_{ij}) for each link of tree network. Then, we use the obtained flow as an input for the following LP model.

Linear Programming Model (Model 7): In this model, we solve our mathematical model using the values of f_{ij} and a_{ij} as the input data. The links a_{ij} are obtained from the outer level, and the flows f_{ij} have been obtained from solving the system of linear equations (Procedure 1). Therefore, we solve the mathematical model (Model 7) to minimise the total cost using the constraints (5.5), (5.6), (5.7), and (5.9). The mathematical model (Model 7) is presented below.

$$\text{(Model 7)} \begin{cases} \text{Minimise} \quad \sum_{i\in N}\sum_{j\in N}\sum_{d\in D} C(d) l_{ij}^{d} \\ \text{Subject to:} \\ a_{ij}(\pi_i - \pi_j) = \beta \sum_{d\in D} l_{ij}^{d} d^{-5} f_{ij}^{2}, & \forall (i,j) \in A \\ \sum_{d\in D} l_{ij}^{d} = a_{ij} L_{ij}, & \forall (i,j) \in A \\ \underline{\pi_i} < \pi_i < \bar{\pi}_i, & \forall i \in N \\ l_{ij}^{d} \geq 0, & \forall (i,j) \in A, \forall d \in D. \end{cases}$$

The model is a linear problem due to the linearity of all constraints. The results of solving this model are in the variables of π_i and l_{ij}^{d} and also presenting the minimal cost. We note that, the nonlinear equation (5.5), with fixing the variables of z_{ij} (to parameters a_{ij}) and f_{ij}, is converted to a linear equation with the only variables of π_i and l_{ij}^{d}.

We use the linear solver (CPLEX) in AIMMS to solve linear mathematical model (Model 7). In the following section, we explain the basic of Cross-Entropy Algorithm.

5.2.1 Cross-Entropy Algorithm

The Cross-Entropy method was proposed by Rubinstein (1997) to estimate the probability of rare events in complex stochastic networks. It was soon realised that the Cross-Entropy could be adapted not only for estimating probabilities of rare events but also for solving difficult optimisation problem by implementing a simple modification. This modification is done by transforming the deterministic

optimisation problem into a related stochastic optimisation problem and then using rare event simulation techniques proposed by Rubinstein (1997).

Several recent applications demonstrate the power of the Cross-Entropy method as a generic and practical tool for solving NP-hard problems (Eshragh et al., 2011). The cross-entropy method is an iterative procedure that involves the following two steps.

- Generation of a sample of random data (trajectories, vectors, etc.) according to some random mechanism.
- Updating the parameters of the random mechanism, on the basics of the data collected, to produce a better sample in the next iteration.

The main idea of cross-entropy in the optimisation problem is initialising the probability distribution function over a feasible region (commonly, a uniform distribution). At each iteration, the probability distribution is adapted based on the random sample collected in the previous iteration. The algorithm procedure ideally is to converge to a discrete uniform distribution for the probability matrix that leads to a global optimal solution (Eshragh et al., 2011). Although the cross-entropy method is still evolving, its performance has been promising in many applications (Eshragh et al., 2011; Mardaneh et al., 2015). In addition, Rubinstein and Kroese (2013) proposed a literature review on the application of cross-entropy.

5.3 Implementation

We develop a new Heuristic Algorithm based on Cross-Entropy algorithm as a solution approach for our MINLP problem. We divide the solution strategy into two stages as follows:

The outer stage: We define a procedure for our proposed heuristic algorithm to generate adjacency matrix for variables z_{ij} randomly based on a probability matrix. The selected tree matrix is known as $A = [a_{ij}]_{n \times n}$.

The inner stage: For a given network, it can be performed using any linear programming solver and the output results are π_i, f_{ij}, and l_{ij}^d.

In our implementation we code the heuristic algorithm in JAVA. We connect a programming language (JAVA), a commercial package (AIMMS) and a package

with functions (EXCEL) (see codes in Appendix). The implementation process is as follows:

- Linking between AIMMS package to JAVA programming language is required to implement the iterative procedures between the outer and inner stages of the proposed solution. The AIMMS package receives the matrix of binary variables (tree network) as input from JAVA programming language to solve the inner minimisation and then reports the results (flow, pressure, and pipeline diameters) to JAVA programming language.
- Linking JAVA programming language to EXCEL package with functions is required as JAVA programming language receives input data from EXCEL package with functions and reports the output to EXCEL package with functions.

In terms of the implementation of the outer level of our heuristic algorithm, we define the following:

- A probability distribution function (pdf).
- A procedure that generates tree network based on a probability matrix.

The following section explains generating the tree structure using a probability matrix in more detail.

5.3.1 Generate Tree Structure Using a Probability Distribution Matrix P^t

We introduce the following notation:

A : The adjacency matrix of the tree network with the elements of a_{ij}.

$$a_{ij} = \begin{cases} 1, & \text{if } i \text{ is a serving node for demand node } j \\ 0, & \text{otherwise.} \end{cases}$$

P^t $=[p_{ij}^t]_{n\times n}$, where p_{ij}^t is the probability of serving node j by node i in iteration t.

P_j^t : The jth column of matrix P^t.

We need to initialise the probability distribution matrix P^0. Each column of matrix P^0 is specified for each node j that can be served via any of its candidate

connected nodes with equal likelihood. In other words, p_{ij}^0 in matrix P^0 represents the probability of serving node j by node i, Thus, the elements p_{ij}^0 for each column j of matrix P^0 (P_j^0), is defined as follows:

$$p_{ij}^0 = \begin{cases} 1/(\text{n-1}), & \text{if } j \text{ is a consumer node } (j \in N_c) \qquad \forall i \in N \\ 0, & \text{if } j = 1 \text{ or if } i = j. \end{cases}$$

Consequently, the initial probability matrix P^0 for a number of n nodes is presented below. The n nodes include one source node (node 1) and $n-1$ consumer nodes (nodes 2, 3,..., n)

$$P^0 = [p_{ij}^0]_{n\times n} = \begin{bmatrix} 0 & 1/(n-1) & 1/(n-1) & 1/(n-1) & \dots & 1/(n-1) \\ 0 & 0 & 1/(n-1) & 1/(n-1) & \dots & 1/(n-1) \\ 0 & 1/(n-1) & 0 & 1/(n-1) & \dots & 1/(n-1) \\ 0 & 1/(n-1) & 1/(n-1) & 0 & \dots & 1/(n-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1/(n-1) & 1/(n-1) & 1/(n-1) & \dots & 0 \end{bmatrix}_{n\times n}$$

All candidate serving nodes for each node j (possible nodes to be serving of node j) are defined on each column of P^0 (P_j^0). Moreover, the first column is zero because node 1 is a source node. The sum of all elements of each column in the matrix P^0 (except the first column) equals one because only one serving node must be defined for each node $j \in N_c$.

$$\sum_{i=1}^{n} p_{ij}^0 = 1, \qquad \forall j \in N_c = \{2, 3, \dots, n-1, n\} \tag{5.11}$$

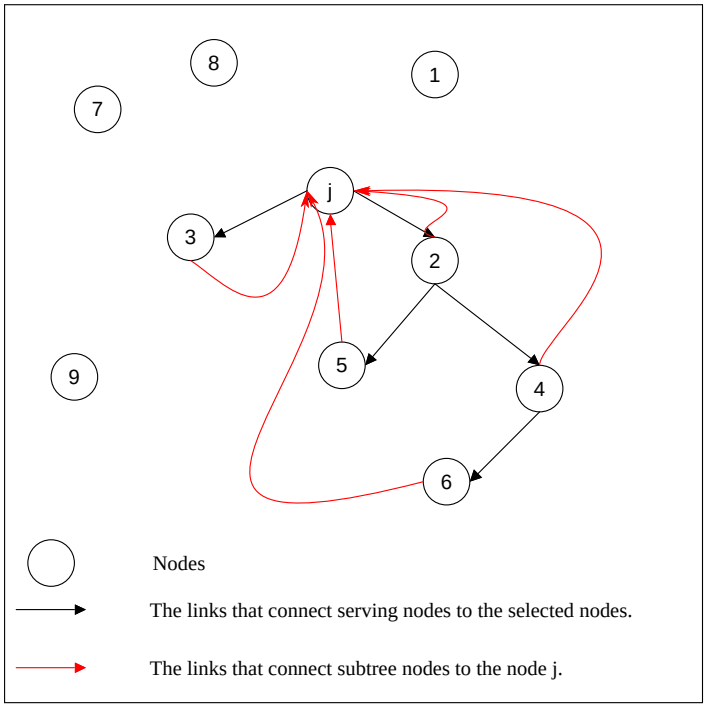

Figure 5.1: An example to generate a tree network with excluding all generation nodes to avoid cycle

where the column probability matrix P^0 for node j is defined as follows:

$$P_j^t = \begin{bmatrix} p_{1j}^t \\ p_{2j}^t \\ \vdots \\ 0 \\ \vdots \\ p_{nj}^t \end{bmatrix} = \begin{bmatrix} 1/(n-1) \\ 1/(n-1) \\ \vdots \\ 0 \\ \vdots \\ 1/(n-1) \end{bmatrix}$$

We note that the probabilities in P^0 are updated whenever a new value for a_{ij} is selected. This is to ensure that our developed algorithm will create a tree network starting at node 1 (source node).

For example, if Consumer 4 is served via Consumer 2, then the probability that Consumer 2 itself is served via Consumer 4 is updated to zero. Moreover, when selecting the value of z_{ij} (looking for the serving node for node j), we need to exclude any node in the sub-tree with root node j to avoid cycles. Figure 5.1 presents an example for generating a tree network that nodes $\{2, 3, 4, 5, 6\}$ have been already served (the black lines). In this example, node j is a node to be served via one of candidate the nodes of the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. We must exclude the nodes from this set which make a cycle with node j. As Fig 5.1 shows, node j is a root for sub-tree including nodes $ST = \{2, 3, 4, 5, 6\}$. The red lines present that if node j is served via any nodes of set ST, a cycle is made, which is unwanted.

As shown in Figure 5.1 for generating tree network, we need to exclude all generation nodes for such a node j from all candidate serving nodes of the node j. The algorithm for generating a tree network using P^t, in the procedure t, is described below.

Algorithm 5: Generate tree structure using the distribution matrix P^t (Procedure (t))

Input : n, P^t, A = $[0]_{n\times n}$.

Step 1 : Set $2 \longrightarrow j$.

Step 2 : Determine the set of G_j where

$G_j = \{i|\ i$ is a node in the sub-tree with root node j$\}$.

Step 3 : $0 \longrightarrow p_{ij}^t$ for all $i \in G_j$.

Step 4 : Normalise column j of P^t:

$$\frac{P_j^t}{\sum_{i=1}^{n} p_{ij}^t} \longrightarrow P_j^t$$

Step 5 : Find Cumulative Distribution Function (CDF) of column j of P^t.

Step 6 : Generate a random number between $[0, 1]$ to find an interval from CDF in step 5.

Step 7 : Determine parent of node j based on the found interval in Step 6.

Step 8 : Set $1 \longrightarrow a_{(parentj)j}$.

Step 9 : If $j < n$ then $j + 1 \longrightarrow j$ and go to Step 2, otherwise return A as the adjacency matrix.

Output : The tree adjacency matrix A.

The explanation of some individual steps of the algorithm are presented below in more detail.

Step 1 : We exclude the supplier node 1 that is the root of the tree network. Therefore, Step 1 is started from node 2 as the first demand node.

Steps 2-3 : We need to exclude any node in the sub-tree with root node j to avoid cycles. Therefore, first of all, we determine all sub-trees of node j (set G_j) to exclude them from serving node (parent) j. Then, we set probability zero to all these nodes that lead to preventing these nodes to be the parent of node j and to avoid cycles.

Step 4 : As some of non zero elements of column P_j^t are set to 0 in Step 3, then, according to equation (5.11), this column should be normalised.

Steps 5-6-7 : In these steps, we determine the CDF of column j of P^t from its pdf. Then, a random number from the uniform distribution of $[0, 1]$ is generated. This random number belongs to an interval of values from the CDF that determines the node to be the parent of node j.

Step 8 : From the previous steps, the parent (serving node) for node j is determined. This means that there is a connection link from the parent node to the node j. Then $a_{(parentj)j} = 1$.

Steps 9 : This procedure will continue until a parent is selected for all demand nodes. Then it terminates and presents the adjacency matrix A that shows a connected tree network.

A flowchart of the algorithm is displayed in Figure 5.2. Also, we present an example for more clarification along with some shapes in Figure 5.3.

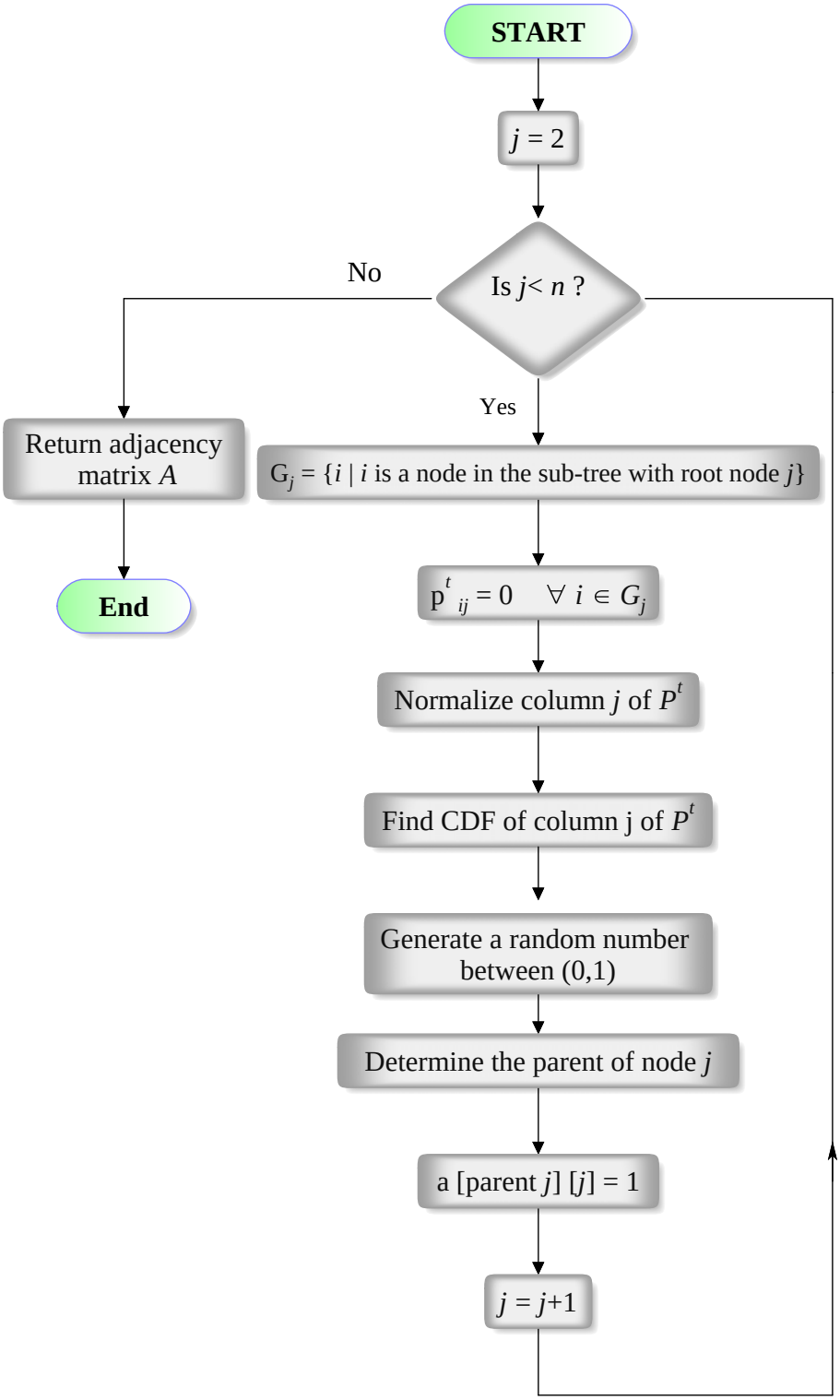

Figure 5.2: Flowchart of Generating the Tree Network

Example for Procedure (t), Generating a tree network

We present an example to generate a tree network with five nodes using Procedure (t). Let P^0 be the initial probability matrix.

$$P^0 = [p^0_{ij}]_{5\times 5} = \begin{bmatrix} 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0.25 & 0 & 0.25 & 0.25 \\ 0 & 0.25 & 0.25 & 0 & 0.25 \\ 0 & 0.25 & 0.25 & 0.25 & 0 \end{bmatrix}_{5\times 5}$$

Iteration 1: $(j = 2)$

1. $j = 2$.
2. Sub-tree of node 2, $G_2 = \emptyset$
3. There is no child for node 2.
4. $P^0_2 = [0.25, 0, 0.25, 0.25, 0.25]$, the column of node 2 of P^0.
5. CDF$(P_2) = [0.25, 0.25, 0.5, 0.75, 1]$.
6. Random between $[0, 1] = 0.4$.
7. $0.25 < 0.4 < 0.5$ then $parent[2] = 3$, $(3 \longrightarrow 2)$.
8. $a_{32} = 1$.
9. $2 < 5$ then $j = 3$ then go to Step 2.

Iteration 2: $(j = 3)$

2. Sub-tree of node 3, $G_3 = \{2\}$.
3. $p^0_{23} = 0$.
4. $P^0_3 = [0.25, 0, 0, 0.25, 0.25]$, the column of node 3 of P^0 then the normalisation is $P^0_3 = [0.33, 0, 0, 0.33, 0.33]$.
5. CDF$(P_2) = [0.33, 0.33, 0.33, 0.66, 1]$.
6. Random between $[0, 1] = 0.9$.

7. $0.66 < 0.9 < 1$, then $parent[3] = 5$, $(5 \longrightarrow 3 \longrightarrow 2)$.
8. $a_{53} = 1$.
9. $3 < 5$ then $j = 4$ then go to Step 2.

Iteration 3: $(j = 4)$

2. Sub-tree of node 4, $G_4 = \emptyset$.
3. There is no child for node 4.
4. $P_4^0 = [0.25, 0.25, 0.25, 0, 0.25]$, the column of node 2 of P^0.
5. $\text{CDF}(P_2^0) = [0.25, 0.5, 0.75, 0.75, 1]$.
6. Random between $[0, 1] = 0.15$.
7. $0 < 0.15 < 0.25$, then $parent[4] = 1$, $(5 \longrightarrow 3 \longrightarrow 2,\ 1 \longrightarrow 4)$.
8. $a_{14} = 1$.
9. $4 < 5$ then $j = 5$ then go to Step 2.

Iteration 4: $(j = 5)$

2. Sub-tree of node 5, $G_5 = \{2, 3\}$.
3. $p_{25} = 0, p_{35} = 0$.
4. $P_5^0 = [0.25, 0, 0, 0.25, 0]$, the column of node 4 of P^0 then the normalisation is $P_5^0 = [0.5, 0, 0, 0, 0.5]$.
5. $\text{CDF}(P_5^0) = [0.5, 0.5, 0.5, 1, 0]$.
6. Random between $[0, 1] = 0.45$.
7. $0 < 0.6 < 0.5$, then $parent[5] = 1$, $(1 \longrightarrow 5 \longrightarrow 3 \longrightarrow 2,\ 1 \longrightarrow 4)$.
8. $a_{15} = 1$.
9. $5 < 5$ then go to Step 10.
10. $$A = [\mathrm{a_{ij}}]_{5\times5} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}_{5\times5}$$

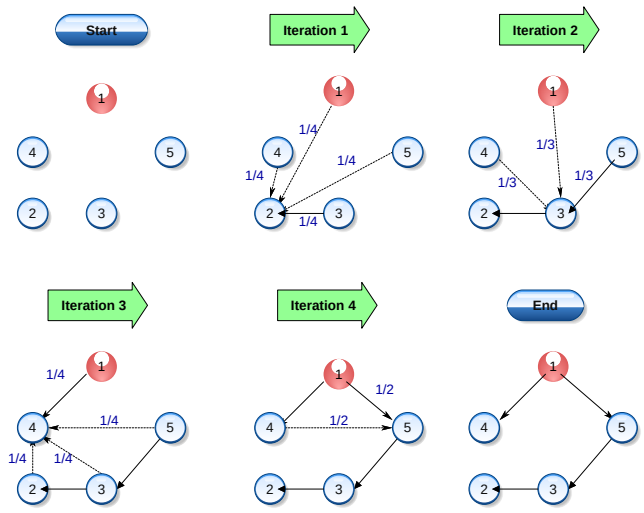

Figure 5.3: Example for Generating of the Tree Network

Figure 5.3 gives an example of Procedure t (Algorithm 5). The output of Procedure t (Algorithm 5) is a set of values a_{ij}, $i = 1, ..., n, j = 1, ..., n$, that define the tree network topology (adjacency matrix A).

5.3.2 Proposed Heuristic Algorithm

The first procedure for our heuristic algorithm is to generate random tree networks. In the second procedure, the cost corresponding to each network is computed by solving the inner stage by AIMMS, the LP solver, and then this information is used to update the probability matrix P^t, with the aim of producing a better tree network in the next iteration. Ideally, P^t, should converge to a certain matrix in which each column has exactly one element equal to 1 except column one (supply node) and all other elements equal to 0. Since $0 \leq p_{ij}^t \leq 1$, we have

$$(p_{ij}^t)^2 \leq p_{ij}^t$$

Hence,

$$\sum_{i=1}^{n} (p_{ij}^t)^2 \leq \sum_{i=1}^{n} p_{ij}^t = 1, \qquad \text{for all } j = 2, \ldots, n \quad (j \in N_c)$$

This implies that the Euclidean norm of the probability matrix P^t satisfies

$$\| P^t \| = \sqrt{\sum_{i=1}^{n} \sum_{j=2}^{n} (p_{ij}^t)^2} \leq \sqrt{n-1}$$

.

Any $n \times n$ certain matrix has the maximum Euclidean norm of $\sqrt{n-1}$ (the first column elements, supply node, equal to zero). Hence, we use the following convergence criterion:

$$\left| \| P \| - \sqrt{n-1} \right| < \varepsilon$$

$$-\varepsilon < \| P \| - \sqrt{n-1} < \varepsilon$$

Where $\varepsilon > 0$ is a given tolerance parameter. Our heuristic algorithm breaks up when this inequality is satisfied, or when the best upper bound for the optimal cost does not change over r iterations. The details of our heuristic algorithm are described below along with the flowchart in Figures (5.4) and (5.5). The input parameters are initialised by giving:

- M : Sample size
- M^{elite} : Elite sample size,
- $\alpha \in [0, 1]$: Smoothing parameter
- ε : Tolerance parameter
- r : An integer parameter
- U^0 : The upper bound for the optimal cost

Algorithm 6: New Heuristic Algorithm

Input : $M, M^{elite}, n, \alpha, \varepsilon, r, U^t, P^0, A_l = [0]_{n\times n}$

Step 1 : Set $\infty \longrightarrow U^0$ and $0 \longrightarrow t$.

Step 2 : Generate initial sample of binary matrix $A_1, A_2, \ldots, A_M$ using Procedure (t) with P^t as the initial distribution matrix. Note that each $A_l; l = 1, \ldots, M$ is a matrix of $a_{ij}; i = 1, \ldots, n; j = 1, \ldots, n$.

Step 3 : For each sample adjacency matrix $A_l; l = 1, \ldots, M$, determine the inner level optimisation problem by solving the resulting linear programming problem in AIMMS. If the tree is infeasible (does not solve), recheck and remove this tree from the sample list and make a new binary matrix. Repeat this cycle to generate a feasible tree. Let $C_1, \ldots, C_M$ denote the optimal costs obtained.

Step 4 : Sort A_l's in ascending order with respect to values of C_l. This creates a sequence of order statistics $C_{(1)}, \ldots, C_{(M)}$ with the property:

$$Pr(C_{(1)} \leq C_{(2)} \leq \cdots \leq C_{(M)}) = 1$$

Now we have this $A_{(1)}, A_{(2)}, \ldots, A_{(M)}$ sorted in ascending order with respect to values of $C_{(j)}$. Then consider the best M^{elite} generated solution from $A_{(1)}, A_{(2)}, \ldots, A_{(M)}$, named the elite sample.

Step 5 : If $C_{(1)} < U^t$ then $U^{t+1} = C_{(1)}$ and $A^* = A_{(1)}$, where A^* presents the current best adjacency matrix of a_{ij} variables; otherwise $U^{(t+1)} = U^{(t)}$.

Step 6 : If $t \geq r$ and $U^{(t+1)} = U^{(t)} = U^{(t-1)} = \cdots = U^{(t-r)}$ then STOP and claim that A^* is a near optimal adjacency matrix of z_{ij} variables, otherwise, go to Step 7.

Step 7 : If $\left| \| P \| - \sqrt{n-1} \right| < \varepsilon$ then STOP and claim that A^* is an optimal adjacency matrix of a_{ij} variables, otherwise, go to Step 8.

Step 8 : Use the elite sample to update the probability distribution matrix by applying the following equations:

$$\tilde{p}_{ij}^{t+1} = \frac{v_{ij}}{M^{elite}},$$

$$p_{ij}^{t+1} = (1-\alpha)\ p_{ij}^{t} + \alpha\ \tilde{p}_{ij}^{t+1},$$

where, v_{ij} is the number of times among elite sample such that the variable a_{ij} take the value of 1.

Step 9 : Normalise each column of P^{t+1}, and, then generate a new sample using the updated probability matrix P^{t+1}, set $t = t + 1$ and go to Step 2.

Output : Minimum Cost, f_{ij}, π_i, l_{ij}^d

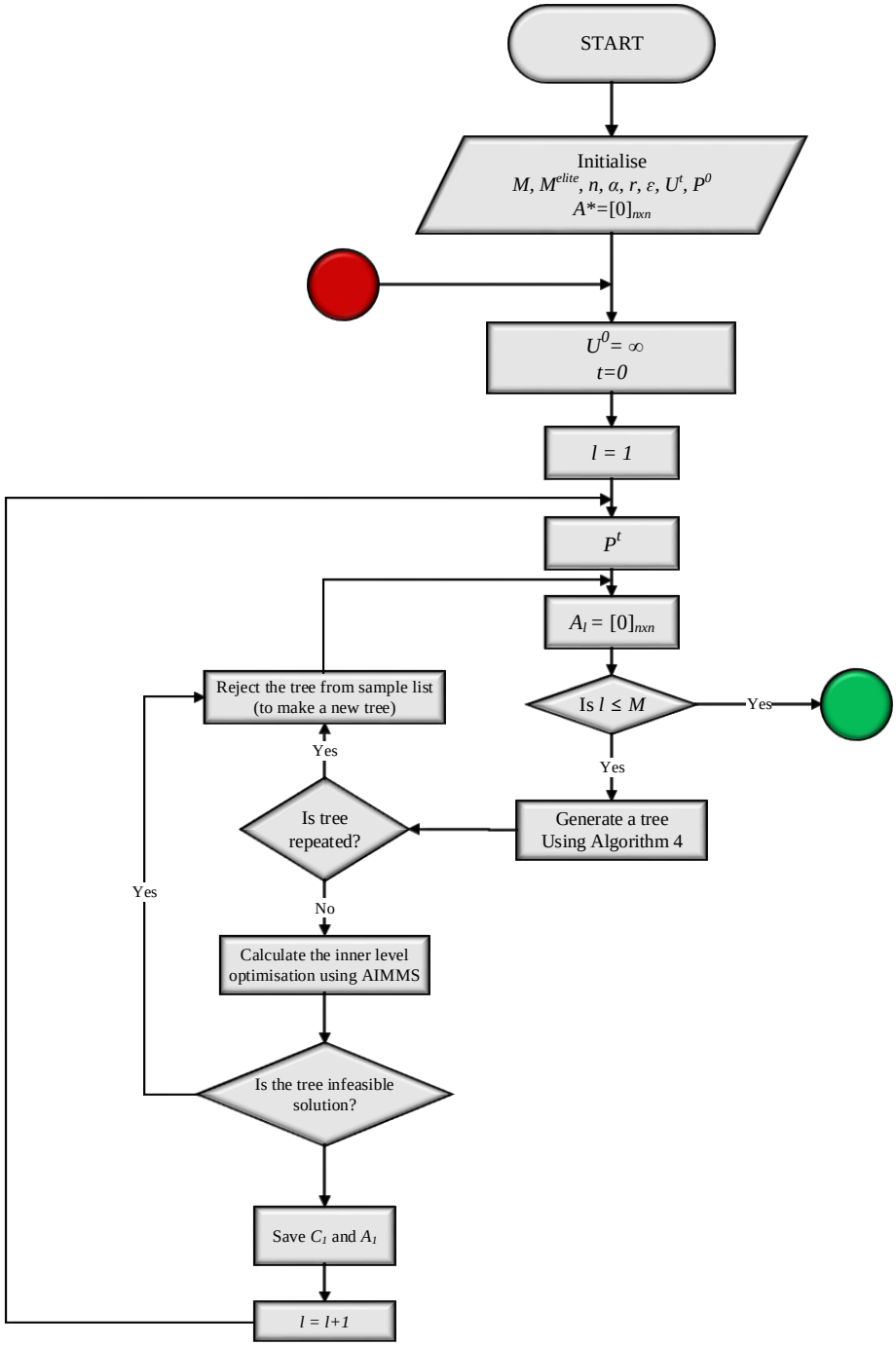

Figure 5.4: Flowchart of Heuristic Algorithm

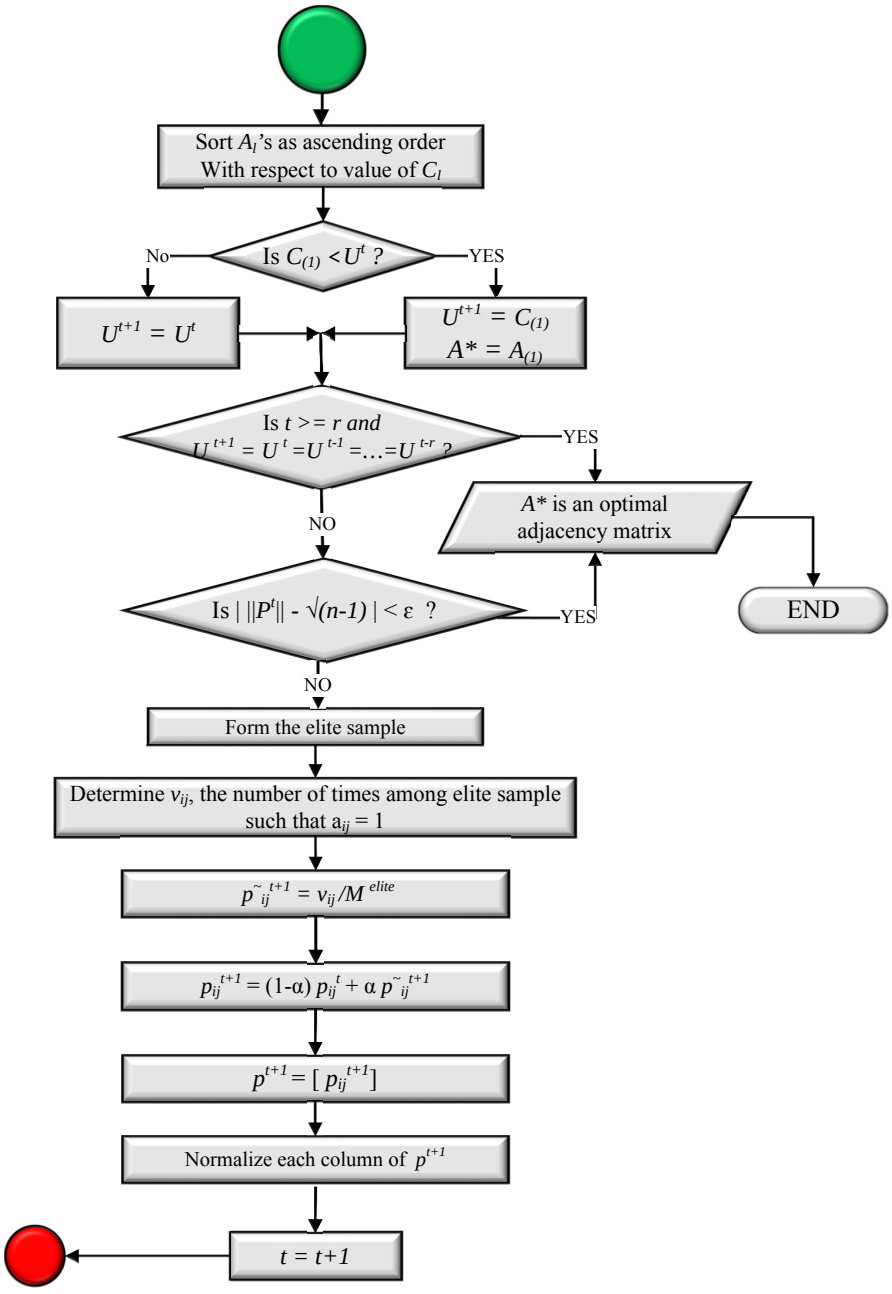

Figure 5.5: Flowchart of Heuristic Algorithm (Continuation of Figure 5.4)

The explanation of some individual steps of the algorithm is presented below in more detail.

Step 2 : We use the initial probability matrix as suggested by Rubinstein and Kroese (2013) to generate samples. We use Algorithm 5 to generate a tree network using probability matrix. This algorithm updates the probability of each node j to exclude any node in the unique path starting at node j to avoid cycles.

Step 3 : The mixed integer nonlinear programming model is simplified to a nonlinear programming model when we fix z_{ij} variables. Generating a tree structure, the result is a linear programming model. The flow variables are obtained iteratively using the system of equations (Procedure 1) in AIMMS, and other variables also obtained from the linear sub-problem (Model 7) using AIMMS-Linear programming solver (CPLEX). If the tree is infeasible (does not solve), recheck and remove this tree from the sample list and make a new binary matrix. Repeat this cycle to generate a feasible tree.

Step 4 : As the objective function is minimised, we sort the performance of each sample by the value found for the linear programming objective function. We select the best elite sample to drive for the current iteration.

Steps 5-7 : The stopping criterion is checked on these steps. If the value of the best performing objective function is better than the current upper bound, the upper bound will be updated, otherwise, it remains the same. Then, the upper bound is compared to some previous iteration's upper bound, if any improvement has been made, more investigation is needed, so the algorithm proceeds to Step 7. If upper bound has not changed for the last r iteration, then stop and report that A^* is a near optimal adjacency matrix.

Step 8 : In this step, the probability matrix is updated based on the elite sample. For each z_{ij} variable, a secondary pdf $\tilde{P}$ is determined by evaluating this $\frac{v_{ij}}{M^{elite}}$ where, v_{ij} is the number of times among elite sample such that the variable a_{ij} take the value of 1. Then, the main probability matrix is obtained by using a smoothing parameter to incorporate the historical matrices into play. One desirable consequence of using smoothing parameter is to reduce the probability that some components of the probability matrix P^{t+1} will be zero or one in the early iteration.

Step 9 : In this step, we normalise each column of P^{t+1}, and, then set $t = t + 1$ and go to Step 2.

5.3.3 Interface Between JAVA programming language, AIMMS package, and EXCEL package with functions

We code the heuristic algorithm in JAVA programming language and then connect JAVA, AIMMS package, and EXCEL package with functions together to implement the algorithm (See Java code in Appendix). The implementation process is as follows:

- Linking between AIMMS package and JAVA programming language is required to implement the iterative procedures between the inner level and outer level of our heuristic algorithm (Algorithm 6). AIMMS package receives each sample of the matrix A_l (a tree network that is generated with Procedure t (Algorithm 5)) as input to solve the inner minimisation and then reports the output results (the flow, the pressure, and the pipeline diameters) to JAVA programming language.

- Linking JAVA programming language and EXCEL package with functions is required. JAVA programming language receives input data from EXCEL package with functions and reports the output to EXCEL package with functions.

We have tested our algorithm on the test cases data given in Section 4.1. Our test data is for $n = 5$, 10, 15, 20, and 25. The results show that the algorithm works well. Some instances are terminated with the stopping criteria in Step 5 and other ones with the one in Step 7. Table 5.1 shows the minimum, the average, and the maximum computation time for $n = 10$, 15, 20, and 25.

Table 5.1 illustrates that the computation time is increased significantly, while the network size is increased. We need to consider that the average computation time for networks with 25 nodes is an enormous value. So we need to improve our algorithm.

Table 5.1: The computation times of our algorithm for networks with different nodes

	Time (sec)		
n	Minimum	Average	Maximum
10	467.00	765.33	1,018.00
15	3,670.00	6,624.63	10,665.00
20	5,860.00	10,143.80	14,679.00
25	10,553.00	14,891.23	22,439.00

The comparison of objective function values between our heuristic algorithm and the results of the AOA algorithm for larger size data, 25 nodes, have not been satisfactory. The reason is that our algorithm did not compare well to the NLP-Single start method. We present Table 5.2 for the network with 25 nodes to show the comparison results between our heuristic algorithm and the NLP-Single start algorithm with ROT(MILP)=0.05.

$$\text{Improvement} = \frac{\{\text{The NLP-Single Start}\} - \{\text{Our heuristic algorithm}\}}{\text{Best value}} \times 100\% \tag{5.12}$$

The best value is the minimum value of the results between our heuristic algorithm and the NLP-Single start. Also, the positive and negative value shows improvement and deterioration value in both cost and computation time, respectively for our heuristic algorithm based on the results with the NLP Single-start.

Table 5.2 shows that our algorithm does not present the cheaper cost for all test cases. There is no improvement in the cost for 10 cases out of 30. We also note the results for $n = 10$, 15, and 20 shows that there are some test cases that have not improved in the cost.

Table 5.2: The comparison results in the objective function value between our heuristic algorithm and the NLP Single-start (25 nodes)

Instance	Minimum cost%
1	16.34
2	27.30
3	56.50
4	2.69
5	7.62
6	16.36
7	3.17
8	10.01
9	31.22
10	-15.01
11	-2.16
12	7.19
13	9.73
14	24.45
15	67.33
16	-10.49
17	160.09
18	8.25
19	-28.46
20	-2.05
21	2.83
22	6.80
23	6.57
24	7.12
25	-46.81
26	-53.78
27	-6.27
28	-27.22
29	7.82
30	-3.35

We discuss our heuristic's drawbacks in Section 5.3.4. Then, we present some improvements.

5.3.4 Algorithm's Drawbacks

We outline the drawbacks of our heuristic below resulted due to our choices of initial probability matrix, the number of M samples and the ε value.

Drawback 1 : The initial probability matrix P^0 has an important role in guiding the algorithm to choose better samples. Starting with the same probability

for nodes to be the parent of a node, may not generate a good design.

Drawback 2 : We generate all samples of tree networks. To do this, we consider a big sample size number M. Therefore, it takes time to generate the number of M samples for each iteration.

Drawback 3 : Considering the small value (10^{-4}) for ε increases the computation time. As the value of ε is small, mostly, the algorithm should stop with the first criteria. As a result, the algorithm terminates with two or more repeated iterations without any specific improvement in the result.

5.3.5 Proposed Alternatives for Drawbacks

We investigate to improve our heuristic algorithm to find a cost and time effective feasible objective function value.

Alternative 1 : Consider the column probability matrix P^0 for node j,

$$P_j^t = \begin{bmatrix} p_{1j}^t \\ p_{2j}^t \\ \vdots \\ 0 \\ \vdots \\ p_{nj}^t \end{bmatrix} = \begin{bmatrix} 1/(n-1) \\ 1/(n-1) \\ \vdots \\ 0 \\ \vdots \\ 1/(n-1) \end{bmatrix}$$

There is the same probability for each node to be the parent of node j. Although each node is placed in different distances of the node j, they have the same probability of being the node j's parent.

It is reasonable to consider for each node j that closer nodes are better candidates to be the parent of that node. The objective function (5.1) is the total cost that is the cost of pipeline diameter multiplied by pipeline's length. Therefore, the cost of design decreases with the shorter network's arcs. We note that that the Minimum Spanning Tree (MST) design, is not the cheapest design for the gas distribution network (André et al. (2013)). Therefore, we consider three categories of candidate nodes to stand as the parent of each node j based on the distance values. These groups are outlined below and are shown in Figure 5.6.

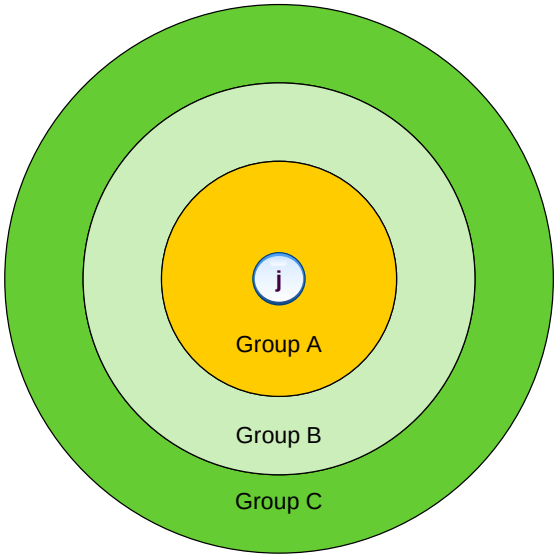

Figure 5.6: Different areas around node j

Group A : This includes the number of nodes that are close to the node j.

Group B : This includes the number of nodes that are neither near nor far to the node j.

Group C : This includes the number of nodes that are far from the node j.

We update the initial probability matrix P^0 with the new definition of elements of p_{ij} based on these three groups. We define below p_{ij}. The probability of nodes to stand as the parent of node j is:

$$p_{ij}^0 = \begin{cases} \frac{n-1}{n}, & \text{for all } i \in A \\ \frac{1}{10n}, & \text{for all } i \in B \\ 0, & \text{for all } i \in C \\ 0, & \text{if } j = 1 \text{ and if } i = j. \end{cases}$$

This leads to an increase of chance of generating better tree networks in our samples. It is reasonable to consider zero probability value for nodes

in group C. This leads to exclude costly tree networks in our samples. We update and normalise each column of the initial probability matrix P^0.

Alternative 2 : We consider a large number sample size $M1$ for the first iteration to generate and, a small sample size $M2$ for other iterations. We save the time regarding the generating sample size problem.

Alternative 3 : We consider a greater value for parameter ε than 10^{-4} for example 10^{-2}.

We test our heuristic algorithm for all 30 instances for categories of 10, 25, 50 nodes again as the small size, the medium size, and the large size, respectively for our model. Our heuristic algorithm finds a cost-effective objective value in a reasonable computation time. The reasonable computation time for our heuristic algorithm is 4 hours. The computational results are reported in the next section.

5.4 Computational Results

In this section, we test three network sizes of the problem including 10 nodes (small size); 25 nodes (medium size); and 50 nodes (large size). Each network includes the 30 instances which have been tested using the AOA algorithm in Chapter 4. Some parameters are to be initialised for our heuristic algorithm for each category. These are outlined below.

- $M1$: Sample size of the first iteration.
- $M2$: Sample size of the other iteration ($M2$ is 10 percent of $M1$).
- $M1^{elite}$: Elite sample size of the first iteration.
- $M2^{elite}$: Elite sample size of the other iteration ($M2^{elite}$ is 10 percent of $M1^{elite}$).
- $\alpha \in [0, 1]$: Smoothing parameter.
- ε : Tolerance parameter ($\varepsilon = 10^{-2}$).
- r : An integer parameter ($r = 5$).
- n : Number of nodes ($|N| = n$).
- U^0 : Initial upper bound for the cost ($U^0 = \infty$).

The value of sample sizes and elite sample sizes and α depend on the number of nodes. For each network size, we determine these values considering the following steps:

- Pick three instances from the uniform distribution of $[1, 30]$.
- Test the algorithm for different values of parameters M, M^{elite}, α. To find a good configuration of these parameters M, M^{elite} and α, we follow the same analysis used by Eshragh et al. (2011); Mardaneh and Caccetta (2016). For each of these parameters, our algorithm is tested based on the following:
  - Consider different values for one parameter and fix the value for other parameters. For example, we consider the different values of M while M^{elite} and α are fixed.
- Compare the results of minimum cost and computation time and the number of iterations on the basis of provided improvement functions (will be detailed further in this Chapter).
- Determine values for the parameters of sample size and elite sample size and α that have a cost-effective solution value with less computation time and less number of iteration compared with the other parameters.
- Test all instances based on these values of sample size and elite sample sizes and α.

5.4.1 10 Node Networks

We start with networks of 10 nodes. We explain the internalising values of the sample size, elite sample size and α in detail as follows:

- We consider different values of sample sizes as $150, 250$, and 500, and fix the value of the parameters $M^{elite} = 50$ and $\alpha = 0.9$. We calculate the average results of 3 instances. In Table 5.3, we show the comparative analysis based on:
  Improvement =

  $$\frac{\{\text{The results of } M1 = 500\} - \{\text{The results of } M1 = 150, 250\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $M1 = 500$ and the results of $M1 = 150$, and 250. Also, the positive

and negative value shows improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1 = 500$.

Table 5.3: Sample sizes analysis (10 nodes)

	Improvement		
Different M1	Minimum Cost%	Time (sec)%	Iteration%
500	0.00	0.00	0.00
250	-2.35	37.50	-244.44
150	0.74	-40.91	0.00

As shown in Table 5.3, there is a small change in the total cost for other values of parameter $M1$ compared with the results of $M1 = 500$. Although, there is a significant difference in the computation time and the number of iterations. Decreasing the value of $M1$ from 500 to 250 results in deteriorations in the minimum cost, number of iteration and improvement in the computation time. In contrast, decreasing the value of $M1$ from 500 to 150, leads to an improvement in the minimum cost and significant deterioration in the computation time. Accordingly, we test all test cases with 10 nodes considering the value 500 for the parameter $M1$.

- We consider different values of the elite sample sizes as $20, 40$, and 50,and fix the value of the parameters $M1 = 500$ and $\alpha = 0.9$. The comparative analysis is based on:
  Improvement =

$$\frac{\{\text{The results of } M1^{elite} = 50\} - \{\text{The results of } M1^{elite} = 20, 40\}}{\text{Best value}} \times 100\%$$

The best value is the minimum value of the results between the results of $M^{elite} = 50$ and the results of $M1^{elite} = 20, 40$. Also, the positive and negative value shows improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1^{elite} = 50$.

Table 5.4: Elite sample sizes analysis (10 nodes)

	Improvement		
Different $M1^{elite}$	Minimum Cost%	Time (sec)%	Iteration%
50	0.00	0.00	0.00
40	-0.01	-81.82	-750.00
20	0.74	-36.36	-122.22

Table 5.4 depicts, decreasing the value of parameter elite sample size, the number of iteration and the computation time increase. Decreasing the value of $M1^{elite}$ from 50 to 40, results in deteriorations in the minimum cost, computation time and number of iteration. Also, decreasing the value of $M1^{elite}$ from 50 to 200 leads to an improvement in the minimum cost and significant deterioration in the computation time and number of iteration. Accordingly, we test all test cases with 10 nodes considering the value 50 for the parameter $M1^{elite}$.

- We consider different values of parameter α as $0.9, 0.5, 0.1$, and fix $M1 = 500$ and $M1^{elite} = 50$. The comparative analysis is based on:
  Improvement =

$$\frac{\{\text{The results of } \alpha = 0.9\} - \{\text{The results of } \alpha = 0.5, 0.1\}}{\text{Best value}} \times 100\%$$

The best value is the minimum value of the results between the results of $\alpha = 0.9$ and the results of $\alpha = 0.5, 0.1$. Also, the positive and negative value shows improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $\alpha = 0.9$.

Table 5.5: α analysis (10 nodes)

	Improvement		
Different α	Minimum Cost%	Time (sec)%	Iteration%
0.9	0.00	0.00	0.00
0.5	0.74	-63.64	-350.00
0.1	0.74	-186.36	-233.33

Table 5.5 presents, decreasing the value of parameter α, results in the computation time and the number of iterations increase. Decreasing the value of α from 0.9 to 0.5 and then to 0.1, leads to an improvement in the minimum

cost and significant deterioration in the computation time and number of iterations. Accordingly, we test all test cases with 10 nodes considering the value 0.9 for the parameter α.

Based on the results from Tables 5.3, 5.4, and 5.5, we test our all instances of network with 10 nodes using the following value of parameters:

First sample size $M1 = 500$,
Second sample size $M2 = 50$,
First elite sample size $M1^{elite} = 50$,
Second elite sample size $M2^{elite} = 5$,
$\alpha = 0.9$,
$\varepsilon = 0.01$.

In Table 5.6, we present the results of our heuristic algorithm.

Table 5.6: Our modified heuristic algorithm results (10 nodes)

Instance	Minimum cost	Time (sec)
1	31,381.21	660.00
2	36,806.16	767.00
3	46,357.88	640.00
4	42,134.17	699.00
5	49,651.17	830.00
6	64,588.00	728.00
7	41,269.26	746.00
8	52,546.30	791.00
9	66,836.52	784.00
10	61,234.47	719.00
11	72,272.65	824.00
12	91,846.64	722.00
13	127,957.58	704.00
14	152,860.46	617.00
15	187,938.60	726.00
16	52,012.31	892.00
17	63,133.17	1,018.00
18	76,519.98	998.00
19	39,623.24	718.00
20	45,010.38	801.00
21	58,548.08	726.00
22	41,106.94	467.00
23	50,563.19	778.00
24	76,312.96	874.00
25	71,588.19	714.00
26	83,856.53	1,004.00
27	116,949.00	1,004.00
28	48,042.21	633.00
29	55,876.63	625.00
30	66,663.86	751.00

The results demonstrate that the modified heuristic algorithm requires less computation time than our new heuristic algorithm. The average computation time of the modified heuristic algorithm is 765.33 compared to 2400 seconds. This represent an improve within the time of about 213.59%. Also, the objective function value is improved with our new algorithm.

5.4.2 25 Node Networks

We consider different values of parameters $M1, M1^{elite}$, and α. We test three test instances using our algorithm. We determine the value of each these parameters to find a good feasible solution in a reasonable time and a few number of iterations. The computational results in the average of minimum cost, computation time, and the number of iterations are presented below.

- We consider different values of parameter M as 400, 1000, 1500, and 2000, and fix the value of other parameters $M1^{elite} = 100$ and $\alpha = 0.9$. The comparative analysis is based on:
  Improvement =

$$\frac{\{\text{The results of } M1 = 1000\} - \{\text{The results of } M1 = 400, 1500, 2000\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $M1 = 1000$ and $M1 = 400, 1500, 2000$. Also, the positive and negative value show an improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1 = 1000$.

Table 5.7: Sample sizes analysis (25 nodes)

	Improvement		
Different M1	Minimum Cost%	Time (sec)%	Iteration%
400	-11.68	171.11	57.14
1000	0.00	0.00	0.00
1500	-0.45	-50.00	10.00
2000	-0.01	-46.72	22.22

We compare the parameter $M1$ with the value of 1000 to other values of 400, 1500 and 2000, in Table 5.7. Considering the value of 400 for the sample size, which is less than 1000 results in an unsatisfactory objective value and a decrease in the computation time and the number of iterations. Also, considering a larger value of sample size than 1000 leads to an increase in the computation time and no significant improvement in the objective value and the number of iterations. Accordingly, to test all test cases with 25 nodes, the reasonable sample size is $M1 = 1000$.

- We consider different values of parameter elite sample size as 100, 200, and 400, and fix the value of parameters $M1 = 1000$ and $\alpha = 0.9$. The comparative analysis is based on:
  Improvement =

$$\frac{\{\text{The results of } M1^{elite} = 100\} - \{\text{The results of } M1^{elite} = 200, 400\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $M^{elite} = 100$ and the results of $M1^{elite} = 200, 400$. Also, the positive and negative value shows improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1^{elite} = 100$.

Table 5.8: Elite sample sizes analysis (25 nodes)

	Improvement		
Different $M1^{elite}$	Minimum Cost%	Time (sec)%	Iteration%
100	0.00	0.00	0.00
200	-0.12	-11.48	-27.27
400	-7.80	-40.16	-90.91

  We compare the results of different values of elite sample sizes with the result of $M1^{elite} = 100$ that is 10% of the fixed sample size. Table 5.8 shows, increasing the value of parameter elite sample size, results in an increase in the computation time and the number of iterations and no significant improvement in the minimum cost.

- We consider different value of parameter α as 0.9, 0.5, and 0.1, and fix the value of parameters $M1 = 1100, M1^{elite} = 100$. The comparative analysis is based on:
  Improvement =

$$\frac{\{\text{The results of } \alpha = 0.9\} - \{\text{The results of } \alpha = 0.5, 0.1\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $\alpha = 0.9$ and $\alpha = 0.5, 0.1$. Also, the positive and negative value shows an improvement and deterioration in the cost, the computation time, and the

number of iteration value based on the result of $\alpha = 0.9$.

Table 5.9: α analysis (25 nodes)

	Improvement		
Different α	Minimum Cost%	Time (sec)%	Iteration%
0.9	0.00	0.00	0.00
0.5	2.27	-65.75	-72.73
0.1	-30.47	-43.09	-54.55

Table 5.9 presents the comparison of results of different values of α based on $\alpha = 0.9$. Considering the smaller value of α than 0.9, results in a significant increase in the computation time and the number of iterations and the deterioration in the minimum cost. Accordingly, to test all test cases with 25 nodes, we consider the value of parameter as $\alpha = 0.9$.

Based on the results in Tables 5.7, 5.8, and 5.9, we test all instances of a network with 25 nodes using the following value of parameters:
First sample size $M1 = 1000$,
Second sample size $M2 = 100$,
First elite sample size $M1^{elite} = 100$,
Second elite sample size $M2^{elite} = 10$,
$\alpha = 0.9$,
$\epsilon = 0.01$.
Table 5.10 presents the computation results of the minimum cost and the computation time for 30 test networks with 25 nodes.

Table 5.10: Our new modified heuristic algorithm results (25 nodes)

Instance	Minimum cost	Time (sec)
1	59,662.29	4,970.00
2	77,859.,12	4,953.00
3	97,102.35	4,659.00
4	81,613.71	5,109.00
5	109,601.78	5,250.00
6	139,932.48	5,131.00
7	72,762.47	5,581.00
8	98,642.77	6,352.00
9	121,036.68	3,386.00
10	108,955.45	3,473.00
11	159,024.36	2,767.00
12	200,426.82	3,101.00
13	211,874.96	5,341.00
14	301,281.16	5,003.00
15	381,399.90	5,287.00
16	68,180.06	4,194.00
17	83,289.63	4,538.00
18	99,049.08	4,715.00
19	55,350.13	4,471.00
20	67,711.79	3,735.00
21	76,657.30	4,080.00
22	92,027.63	3,695.00
23	120,952.12	3,505.00
24	140,258.61	3,488.00
25	97,579.76	6,039.00
26	128,860.30	3,239.00
27	153,369.06	3,382.00
28	83,704.61	3,319.00
29	106,300.73	4,138.00
30	133,378.06	3,643.00

In the next section, we present the computational results for all test cases of networks with 50 nodes.

5.4.3 50 Node Networks

We consider different values of parameters $M1, M1^{elite}$, and α. We test three test instances using our algorithm. We determine the value of these parameters to find a cost-effective feasible solution in a reasonable time and a few number of iterations. The computational results in the average of minimum cost, computation

time, and the number of iterations are presented below.

- We consider different value of the parameter sample sizes as 500, 1000, 1500, and 2000, and fix the value of parameters $M1^{elite} = 100$ and $\alpha = 0.9$. The comparative analysis is based on:

  Improvement =

  $$\frac{\{\text{The results of } M1 = 1000\} - \{\text{The results of } M1 = 500, 1500, 2000\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $M1 = 1000$ and $M1 = 500, 1500, 2000$. Also, the positive and negative value shows an improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1 = 1000$.

Table 5.11: Sample sizes analysis (50 nodes)

	Improvement		
Different M1	Minimum Cost%	Time (sec)%	Iteration%
500	-27.22	88.56	6.67
1000	0.00	0.00	0.00
1500	-13.80	-13.26	0.00
2000	-10.38	-56.74	23.08

Table 5.11 compares the parameter $M1$ with the value of 1000 to other values of 500, 1500 and 2000. Consider the value of 500 for the sample size, which is less than 1000, results in a decrease in the computation time and the number of iterations and a worse objective function value. Also, consider a larger value of sample size than 1000, there is an increase in the computation time and no significant improvement in the objective function and the number of iterations. Accordingly, we test our test cases considering the sample size of $M1 = 1000$.

- We consider different values of parameter elite sample size as 100, 200, and 400, and fix the value of parameters $M1 = 1000$ and $\alpha = 0.9$. The comparative analysis is based on:

Improvement =

$$\frac{\{\text{The results of } M1^{elite} = 100\} - \{\text{The results of } M1^{elite} = 200, 400\}}{\text{Best value}} \times 100\%$$

The best value is the minimum value of the results between the results of $M^{elite} = 100$ and the results of $M1^{elite} = 200$, and 400. Also, the positive and negative value shows an improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $M1^{elite} = 100$.

Table 5.12: Elite sample sizes analysis (50 nodes)

	Improvement		
Different $M1^{elite}$	Minimum cost%	Time (sec)%	Iteration%
100	0.00	0.00	0.00
200	-6.54	-17.75	-33.33
400	-9.52	-70.34	-100.00

We compare the results of different values of elite sample sizes with the result of $M1^{elite} = 100$ that is 10% of the fixed sample size. Table 5.12 shows, increasing the value of parameter elite sample size, results in an increase in the computation time and the number of iterations and no significant improvement of the solution. Accordingly, we test all test cases with 50 nodes considering the value parameter $M1^{elite} = 100$.

- We consider different value of parameter α as 0.9, 0.5, and 0.1, and fix the value of parameters $M1 = 1100$, and $M1^{elite} = 100$. The comparative analysis is based on:

  Improvement =

  $$\frac{\{\text{The results of } \alpha{=}0.9\} - \{\text{The results of } \alpha{=}0.5,\ 0.1\}}{\text{Best value}} \times 100\%$$

  The best value is the minimum value of the results between the results of $\alpha = 0.9$ and $\alpha = 0.5, 0.1$. Also, the positive and negative value shows an improvement and deterioration in the cost, the computation time, and the number of iteration value based on the result of $\alpha = 0.9$.

Table 5.13: α analysis (50 nodes)

	Improvement		
Different α	Minimum Cost%	Time (sec)%	Iteration%
0.9	0.00	0.00	0.00
0.5	-7.64	-44.95	-106.25
0.1	-105.29	52.57	100.00

Table 5.13 shows deterioration in the objective value for the values of α=0.1 and 0.5 compared to α=0.9. For the network with 50 nodes, the cost and time-effective feasible value for α is 0.9.

Based on the results in Tables 5.11, 5.12, and 5.13, we test all instances of network with 25 nodes using the following value of parameters:
First sample size $M1 = 1000$,
Second sample size $M2 = 100$,
First elite sample size $M1^{elite} = 100$,
Second elite sample size $M2^{elite} = 10$,
$\alpha = 0.9$,
$\varepsilon = 0.01$.

The computational results in the minimum cost and computation time for all 30 test cases of network with 50 nodes are reported in Table 5.14.

Table 5.14 shows that the networks with 50 nodes are terminated around 10,000 seconds, and there are no significant changes in time between all 30 instances. Table 5.15 presents the Minimum, Average and Maximum of the time of 30 instances considering three network sizes with 10, 25, and 50 nodes.

Table 5.14: Our new modified heuristic algorithm results (50 nodes)

Instance	Minimum cost	Time (sec)
1	193,491.54	9,377.00
2	131,954.91	10,400.00
3	175,870.17	10,514.00
4	156,144.05	9,761.00
5	211,589.03	9,526.00
6	281,024.12	9,526.00
7	127,408.91	8,771.00
8	180,771.59	8,539.00
9	240,103.60	11,364.00
10	200,469.79	12,371.00
11	290,636.50	8,606.00
12	388,548.28	8,417.00
13	399,770.73	13,411.00
14	552,459.86	11,788.00
15	758,492.83	10,148.00
16	115,537.92	10,562.00
17	127,550.05	11,399.00
18	146,205.80	11,179.00
19	113,364.32	10,195.00
20	148,180.79	9,785.00
21	170,551.30	11,351.00
22	126,824.01	10,240.00
23	162,630.91	8,721.00
24	203,197.49	10,360.00
25	142,085.41	10,948.00
26	194,635.45	9,489.00
27	224,305.68	9,914.00
28	146,889.79	9,495.00
29	196,517.76	9,330.00
30	237,069.79	9,389.00

Table 5.15: The computation times using our new modified heuristic algorithm for networks with 10, 25, 50 nodes

	Time (sec)		
n	Minimum	Average	Maximum
10	467.00	765.33	1,018.00
25	2,767.00	4,351.46	6,352.00
50	8,417.00	10,162.53	13,411.00

Table 5.15 shows that increasing the network size to the double size (from 25 to 50), leads to a progressive increase in the average of computation time. In the next section, we compare the objective value and computation time between the OA/ER/AP algorithm (AOA) and our new modified heuristic algorithm.

5.5 Comparative Analysis (The NLP-Single Starting Point)

In this section, we provide the tables to show the improvement of our new modified heuristic algorithm by percentage, in terms of the minimum cost and computation time based on the results of the AOA algorithm. We determine the improvement based on the following equation:

$$\text{Improvement} = \frac{\{\text{The NLP-Single start results}\} - \{\text{Our modified heuristic results}\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values between the values of the NLP-Single start and our modified heuristic algorithm. Compared with the results of the NLP-Single start algorithm, the positive and negative value show an improvement and deterioration in cost and time for our heuristics algorithm respectively.

We compare our heuristic results with the NLP-Single start condition. We demonstrate that our heuristic algorithm works well in terms of objective function value compared with the NLP-Single start.

5.5.1 10 Node Networks

Table 5.16 compares the minimum cost and the computation time for all 30 test instances between the AOA algorithm with the NLP-Single starting point and our modified heuristic algorithm. The results show that our algorithm works well mostly in terms of the minimum cost. However, in terms of the computation time, the AOA performs quicker.

Table 5.16: Improvement of our modified heuristic algorithm compared with NLP-Single start (10 nodes)

Instance	Min cost%	Time (sec)%
1	2.00	-8,747.00
2	3.00	-2,7490.00
3	1.00	-3,274.00
4	-5.00	-5,6271.00
5	7.00	-3,931.00
6	4.00	-6,641.00
7	-5.00	-1,3150.00
8	-5.00	-3,329.00
9	5.00	-2,516.00
10	-2.00	-1,4166.00
11	7.00	-4,251.00
12	9.00	-2,364.00
13	2.00	-3,955.00
14	0.00	-5,265.00
15	23.00	-2,634.00
16	1.00	-2,4205.00
17	1.00	-7,986.00
18	0.00	-6,474.00
19	0.00	-1,9464.00
20	7.00	-2,2851.00
21	-3.00	-5,807.00
22	0.00	-1,0735.00
23	0.00	-7,790.00
24	0.00	-3,363.00
25	5.00	-1,7230.00
26	5.00	-1,2545.00
27	9.00	-6,348.00
28	0.00	-1,8518.00
29	0.00	-9,173.00
30	0.00	-3,370.00

Table 5.17 shows the number of test instances (out of 30) for a network with 10 nodes without/with a change in the cost by our heuristic algorithm.

Table 5.17: Number of instances without/with a change in the cost by our heuristic algorithm (10 nodes)

Deterioration	No Change	Improvement
5	9	16

Table 5.17 illustrates that both algorithms have the same cost for 9 instances out of 30. Compared with the NLP-Single start, results of our heuristic algorithm show a better computation time and an improvement in the cost for half of 30 instances.

5.5.2 25 Node Networks

Table 4.5 compares our modified heuristic algorithm results with the NLP-Single start considering different values of ROT(MILP).

Table 5.18 shows that our heuristic algorithm improves the objective function value for all 30 test instances compared with the NLP-Single start with different values of ROT(MILP). Also, our algorithm gives an improvement in the computation time with ROT(MILP) = 0.01. In contrast, the NLP-Single start shows better performance in the computation time with ROT(MILP) = 0.03, and 0.05.

Table 5.18: The comparison results between our modified heuristic algorithm and the NLP-Single start (25 nodes)

	ROT(MILP)=0.05		ROT(MILP)=0.03		ROT(MILP)=0.01	
Instance	Minimum Cost%	Time (sec)%	Minimum Cost%	Time (sec)%	Minimum Cost%	Time (sec)%
1	12.35	-552.76	7.90	-61.47	7.90	8.48
2	43.49	1.47	64.86	-4.87	72.33	20.60
3	60.74	-76.45	60.28	103.57	89.15	455.89
4	5.91	-699.59	4.41	-827.73	4.49	-331.97
5	21.51	-670.31	21.51	-619.35	37.18	17.98
6	31.55	-527.65	31.55	-31.75	31.81	65.03
7	7.99	-1382.81	7.99	-437.74	7.99	-64.93
8	23.64	-253.09	23.64	-90.13	23.64	-83.42
9	40.52	27.78	83.71	441.82	87.12	797.40
10	8.94	-733.03	8.94	-730.92	8.94	-249.40
11	15.53	-470.79	15.53	331.88	15.53	9.42
12	13.27	54.46	13.27	99.48	13.27	406.22
13	25.30	-185.74	36.14	-108.91	28.93	-48.30
14	34.78	-138.62	59.15	347.25	27.53	302.79
15	74.45	-106.04	74.45	52.92	0.74	281.15
16	11.27	-87.35	20.07	-39.00	19.58	-25.19
17	172.01	-535.89	178.41	-913.42	178.41	-536.05
18	48.78	216.34	48.78	-68.39	47.35	200.36
19	13.66	-310.19	15.92	-420.70	15.92	-307.33
20	17.93	7.97	17.93	-10.23	17.93	99.01
21	31.87	236.02	31.87	565.95	31.87	937.79
22	38.10	-11.16	36.46	58.05	12.53	10.48
23	20.45	160.83	20.45	314.97	20.45	668.05
24	33.90	139.69	33.90	795.47	33.90	845.25
25	6.78	-216.54	6.78	-213.27	6.78	-112.28
26	7.03	181.32	7.03	211.27	7.03	143.49
27	53.30	-164.31	13.83	294.36	13.83	489.21
28	12.49	-241.03	2.54	-213.37	4.10	-16.11
29	43.10	-44.40	43.10	0.40	43.10	56.13
30	43.69	-0.67	43.69	151.18	43.69	249.16

Table 5.19 presents the number of instances without/with a change in total cost by our heuristic algorithm compared with the NLP-Single start. Table 5.19 shows that our heuristic algorithm improves the objective function value for all 30 test instances with different values of ROT(MILP). However, the improvement of computation time with our heuristic algorithm is increased, when the value of ROT(MILP) decreases. The results demonstrated that about a third, half and two-thirds of 30 instances are improved in the computation time when the value of ROT(MILP) are 0.05, 0.03, and 0.01, respectively.

Table 5.19: Number of instances without/with a change in the cost by our heuristic algorithm (25 nodes)

Different ROT(MILP)	ROT(MILP) = 0.05		ROT(MILP) = 0.03		ROT(MILP) = 0.01	
Number of instances solved by the AOA	30		30		30	
Objective function value and computation time	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)
Number of instances improved by Heuristic	30	9	30	14	30	20

5.5.3 50 Node Networks

In this section, we compare the results of the heuristic algorithm and the NLP-Single start. We need to note that, this comparison is for instances that have been solved by the NLP-Single start. We consider the results of the NLP-Single start from Tables 5.14 and 4.5 with different values of ROT(MILP). Also, we write 'NS' in Table 5.20, since the AOA does not solve some instances with the NLP-Single starting point.

Table 5.20 indicates that our heuristic algorithm always improves the objective function value. We note that our algorithm finds a cost-effective feasible solution for all 30 test instances of the network with 50 nodes compared with the AOA algorithm. Also, the improvement of computation time with our heuristic algorithm is increased, when the value of ROT(MILP) is decreased. In Table 5.21, we present the number of improved instances in the cost and computation time by our modified heuristic algorithm compared with a number of solved instances by the NLP-Single start.

Table 5.20: The comparison results between our new modified heuristic algorithm and the NLP-Single start 50-node networks. Note that 'NS' means No Solution obtained within the specified time limit.

	ROT(MILP)=0.2		ROT(MILP)=0.15		ROT(MILP)=0.1	
Instance	Minimum Cost%	Time(sec)%	Minimum Cost%	Time (sec)%	Minimum Cost%	Time(sec)%
1	343.64	-1124.55	12.42	-754.37	133.36	132.79
2	19.74	-807.94	27.40	-613.54	34.57	174.30
3	24.05	-1035.07	24.05	-396.47	NS	NS
4	15.66	-1358.35	21.40	-978.24	22.07	9.78
5	40.63	-739.47	40.63	-369.09	36.28	588.82
6	40.58	-523.27	40.58	-128.25	NS	NS
7	10.66	-825.03	16.59	-506.13	5.19	62.84
8	35.98	-655.64	29.49	-229.90	NS	NS
9	44.84	-409.82	49.59	22.11	NS	NS
10	22.00	-1533.00	22.00	-515.00	18.00	-70.00
11	28.00	-810.00	28.00	-247.00	24.00	503.00
12	29.00	-872.00	29.00	-417.00	29.00	159.00
13	23.00	-1452.00	29.00	-358.00	30.00	452.00
14	43.00	-1167.00	43.00	-87.00	NS	NS
15	32.00	-1142.00	32.00	-334.00	32.00	544.00
16	-20.00	-1295.00	-20.00	-75.00	-20.00	-1.00
17	-1.00	-467.00	-1.00	79.00	NS	NS
18	11.00	-395.00	11.00	519.00	NS	NS
19	9.00	-398.00	14.00	15.00	NS	NS
20	22.00	-286.00	22.00	206.00	NS	NS
21	44.00	21.00	NS	NS	NS	NS
22	12.00	-346.00	19.00	-224.00	NS	NS
23	40.00	-202.00	40.00	214.00	NS	NS
24	37.00	-236.00	NS	NS	NS	NS
25	28.00	28.00	-88.00	203.00	NS	NS
26	56.00	13.00	NS	NS	NS	NS
27	65.00	38.00	NS	NS	NS	NS
28	53.00	-143.00	53.00	381.00	NS	NS
29	61.00	-48.00	NS	NS	NS	NS
30	42.00	-330.00	NS	NS	NS	NS

Table 5.21 is summarised as follows: In terms of the objective function value, there is, always, improvement by our heuristic algorithm for all 30 test instances with different values of ROT(MILP). In terms of computation time, our modified heuristic algorithm improves about 4 of 30 instances considering ROT(MILP) = 0.05. This number is increased to 7 out of 24 and 9 out of 12 instances while the value of ROT(MILP) is 0.03 and 0.01, respectively.

Table 5.21: Number of instances without/with a change in the cost by our heuristic algorithm (50 nodes)

Different ROT(MILP)	ROT(MILP) = 0.2		ROT(MILP) = 0.15		ROT(MILP) = 0.1	
Number of instances solved by AOA	30		24		12	
Objective function value and computation time	Minimum Cost	Time (sec)	Minimum cost	Time (sec)	Minimum Cost	Time (sec)
Number of instances improved by Heuristic	28	4	21	7	11	9

Conclusion

In conclusion, Table 5.22 presents that increasing the problem sizes, results in a significant improvement in cost for our modified heuristic algorithm compared with the AOA algorithm with the NLP-Single start.

Table 5.22: The average improvement of our modified heuristic algorithm in the cost compared with the NLP-Single starting point

n	10	25			50		
ROT(MILP) value	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of instances that are solved by the NLP-Single start	30	20	30	30	12	24	30
Minimum cost%	2.00	32.00	34.00	32.00	20.00	25.00	40.00

5.6 Comparative Analysis (The NLP-Multi Starting Points)

In this section, we present the comparison results in the total cost and computation time between the AOA algorithm with NLP-Multi start and our heuristic algorithm. We detemine the improvement based on the following equation:

$$\text{Improvement} = \frac{\{\text{The NLP-Multi start results}\} - \{\text{Our modofied heuristic results}\}}{\text{Best value}} \times 100\%$$

The best value is the minimum values between the values of the NLP-Multi start and our modified heuristic algorithm. Also, compared with the results of the NLP-Multi start algorithm, the positive and negative value show an improvement and deterioration value of our heuristics algorithm in the cost and time respectively.

5.6.1 10 Node Networks

We demonstrate that our modified heuristic algorithm works well, since there is the same cost for some instances compared with the AOA algorithm with NLP-Multi start.

Table 5.23: The comparison results between our modified heuristic algorithm and the NLP-Multi start (10 nodes)

Instance	Min cost%	Time(sec)%
1	0.00	-17547.06
2	6.55	-38834.01
3	5.35	-1759.92
4	-2.59	-22893.42
5	2.63	-27202.63
6	3.77	-4060.00
7	-4.16	-9780.79
8	-4.74	-17208.53
9	5.92	-1981.23
10	-1.83	-9970.03
11	7.18	-3104.98
12	8.69	-95.01
13	2.49	-4935.77
14	0.33	-4712.79
15	21.17	-1828.80
16	1.22	-21762.75
17	1.05	-8900.88
18	0.00	-6898.60
19	0.05	-23993.96
20	8.32	-8606.52
21	-2.56	-5797.64
22	19.66	-1141.69
23	0.00	-6179.26
24	-0.21	-17846.61
25	-1.62	-22142.99
26	5.13	-32182.96
27	25.16	-2399.38
28	0.00	-19619.63
29	0.00	-14401.16
30	0.06	-99.09

Table 5.23 presents the number of instances without/with a change in the cost

between the two algorithms.

Table 5.24: Number of instances without/with a change in the cost by our heuristic algorithm (small size network)

Deterioration	No Change	Improvement
7	5	18

As shown in Table 5.24, using our algorithm, more than half of instances out of 30 have shown improvements in the cost. The NLP-Multi starting point has not performed better than the NLP-Single starting point in the objective function value. As a result, our algorithm find the cheaper design for a network in more computation time.

5.6.2 25 Node Networks

In this part, we present the improvement of our heuristic algorithm in the cost and computation time with NLP-Multi start algorithm's results. Also, we consider different values of the ROT(MILP) for the AOA algorithm with the NLP-Multi starting point.

Table 5.25 presents the number of instances without/with a change in the cost by our heuristic algorithm out of number of solved instances by the NLP-Multi starting points.

Table 5.25: The comparison results between our modified heuristic algorithm and the NLP-Multi start 25-node networks. Note that 'NS' means No Solution obtained within the specified time limit.

	ROT(MILP)=0.05		ROT(MILP)=0.03		ROT(MILP)=0.01	
Instance	Minimum Cost%	Time(sec)%	Minimum Cost%	Time(sec)%	Minimum Cost%	Time(sec)%
1	14	-161	3	-19	8	11
2	42	-76	59	97	72	-25
3	45	-4	55	266	75	330
4	3	-1609	5	-852	4	-700
5	22	-729	22	-566	32	7
6	22	-263	43	-226	43	-33
7	8	-810	8	-282	8	-39
8	24	-233	24	-150	24	-88
9	80	-16	79	68	NS	NS
10	9	-585	9	-112	9	-190
11	16	-390	16	159	16	-19
12	13	-57	13	291	13	406
13	31	-340	12	-137	17	-3
14	75	17	75	374	28	384
15	70	33	74	-9	74	214
16	12	-101	9	-122	-1	-35
17	165	-1067	177	-762	170	-758
18	49	176	49	-54	47	552
19	15	-9534	16	-300	11	-287
20	18	28	18	57	18	225
21	32	225	32	163	32	229
22	13	-246	13	-61	12	56
23	20	90	20	312	20	592
24	34	241	34	395	34	1991
25	7	-70	7	-188	7	-2
26	7	21	7	312	7	387
27	14	155	14	113	14	272
28	17	-248	4	-64	4	-5
29	43	-18	43	-301	43	207
30	44	105	44	-9	44	364

Table 5.26: Number of instances without/with a change in the cost by our modified heuristic algorithm (25 nodes)

Different ROT(MILP)	ROT(MILP) = 0.05		ROT(MILP) = 0.03		ROT(MILP) = 0.01	
Number of instances solved by the AOA	30		30		29	
Objective function value and computation time	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)	Minimum Cost	Time (sec)
Number of instances improved by our heuristic	30	10	30	12	28	16

Table 5.26 presents that our heuristic algorithm works well in the objective function value compared with the NLP-Multi start and with different values of ROT(MILP) for all 30 test instances. The improvement of our modified heuristic algorithm in the computation time is one-third of 30 instances with ROT(MILP) = 0.05. This number is increased to 12 and 16 when the value of ROT(MILP) is 0.03 and 0.01, respectively.

5.6.3 50 Node Networks

In this part, we compare the total cost and the computation time for 30 test instances of networks with 50 nodes between the NLP-Multi start and our modified heuristic algorithms. The 'NS' shows that some instances are not solved using the NLP-Multi start within 24 hours time limit.

Table 5.27: The comparison results between our modified heuristic algorithm and the NLP-Multi start for 50-node networks. Note that 'NS' means No Solution obtained within the specified time limit.

	ROT(MILP)=0.2		ROT(MILP)=0.15		ROT(MILP)=0.1	
Instance	Minimum Cost%	Time (sec)%	Minimum Cost%	Time (sec)%	Minimum Cost%	Time (sec)%
1	8	-715	22	-547	22	131
2	24	-656	30	-259	13	391
3	24	-462	24	-308	24	301
4	5	-739	5	-500	22	-12
5	32	-560	32	-282	30	390
6	38	-555	41	-179	NS	NS
7	11	-662	11	-273	15	-107
8	23	-531	25	-66	NS	NS
9	41	-508	33	46	NS	NS
10	14	-1002	22	-514	22	-24
11	28	-554	28	-350	28	452
12	29	-546	29	-237	29	117
13	26	-942	22	-725	40	259
14	33	-707	44	-41	NS	NS
15	32	-765	32	-295	NS	NS
16	-20	-588	-20	-79	-20	171
17	-1	-312	NS	NS	NS	NS
18	11	67	NS	NS	NS	NS
19	14	-419	14	-35	NS	NS
20	22	-171	NS	NS	NS	NS
21	44	-50	NS	NS	NS	NS
22	74	-264	14	-125	NS	NS
23	35	-210	NS	NS	NS	NS
24	37	-251	NS	NS	NS	NS
25	39	6	NS	NS	NS	NS
26	56	52	NS	NS	NS	NS
27	65	39	NS	NS	NS	NS
28	30	-208	NS	NS	NS	NS
29	43	-79	NS	NS	NS	NS
30	38	23	NS	NS	NS	NS

Table 5.28 presents the number of instances without/with a change in the cost by our modified heuristic algorithm out of the number of solved instances by the NLP-Multi starting points.

Table 5.28: Number of instances without/with a change in the cost by our heuristic algorithm (50 nodes)

Different ROT(MILP)	ROT(MILP) = 0.2		ROT(MILP) = 0.15		ROT(MILP) = 0.1	
Number of instances solved by the AOA	30		18		10	
Objective function value and computation time	Minimum cost	Time (sec)	Minimum cost	Time (sec)	Minimum cost	Time (sec)
Number of instances improved by Heuristic algorithm	30	5	17	1	9	7

Table 5.28 shows that increasing the value of ROT(MILP), results in a decrease in the number of solved instances out of 30 by the NLP Multi start. However, our algorithm performs well in terms of minimum cost for all test instances. This improvement can be outlined as 30 out of 30, 17 out of 18, and 9 out of 10 with ROT(MILP) = 0.02, 0.15, and 0.1 respectively. The AOA algorithm with the NLP-Multi start performs well, mostly, in terms of the computation time with ROT(MILP) = 0.2. In contrast, the computation time is improved by 7 out of 10 with ROT(MILP) = 0.1 by our modified heuristic algorithm.

Conclusion

Table 5.29 shows that increasing the problem sizes, leads to a significant improvement in cost by our heuristic algorithm. We consider the average cost improvement of instances out of solved instances by the AOA algorithm with the NLP-Multi start. Accordingly, our algorithm illustrates better performance in the objective function value than the AOA algorithm with the NLP-Multi start, in particular, for the large sized problems.

Table 5.29: The average improvement of our heuristic algorithm in the minimum cost

n	10	25			50		
ROT(MILP) value	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of instances that are solved by the NLP-Multi start	30	20	30	29	10	18	30
Minimum cost %	3.57	30	33	32	29	23	29

5.7 Conclusion

In this section, we developed, implemented and tested our new heuristic algorithm to solve our mathematical model (Model 6) given by equations (3.12)-(3.20). The quantitative analysis of the effects of different parameters and stopping criteria of our heuristic algorithm on optimal decisions was investigated. The computational results are undertaken considering the effective parameters of sample size, elite sample size and α and the value of ε and r for stopping criteria. Table 5.30 shows the value of parameters, stopping criteria and the average of computation time for each network sizes.

Table 5.30: The value of parameters and the average of computation times for different size network using our heuristic algorithm

n	$M1$	$M2$	$M1^{elite}$	$M2^{elite}$	α	ϵ	r	Average time (sec)
10	500	50	50	5	0.9	0.01	5	765.33
25	1000	100	100	10	0.9	0.01	5	4,351.46
50	1000	100	100	10	0.9	0.01	5	10,162.53

Table 5.30 shows that increasing the network size also increases the computation time. For each network size, we considered the same value for α, ϵ and r. As above table shown, for a network with 25 and 50 nodes, the number of sample size is increased into 1000.

Solving our model (Model 6) with two different methods (approximation and heuristic method), we compared the performances of our heuristic algorithm and the AOA algorithm in terms of objective values and computation times for all test cases.

We restate two Tables 5.31 and 5.32 again that represent the average improvement in the cost and computation time of our heuristic algorithm than the approximation method among all solved test cases for $n = 10, 25$, and 50. These two tables show that our heuristic algorithm performs well in the objective value with different sizes of the network compared with the AOA algorithm considering different values of ROT(MILP) under the NLP-Single start and the NLP-Multi start modules.

Furthermore, the following results can be seen from Tables 5.31 and 5.32.

- Our heuristic algorithm improves the objective function value for small size test problems, 2% and 3.57% on the NLP-Single start and NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for medium size test problems, 33% and 28% on the NLP-Single start and the NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for large size test problems, 32% and 27% on the NLP-Single start and the NLP-Multi start, respectively.

Table 5.30 shows an average computation time (seconds) with our heuristic algorithm for $n = 10, 25$, and 50 of 765.33, 4,351.46, and 10,162.53 seconds, respectively. Tables 5.31 and 5.32 show the comparative analysis in the computation time for the 10 nodes test problem in which the OA/ER/AP algorithm finds a feasible solution in a computation time less than our heuristic algorithm. Also, for $n = 25$, and 50, our heuristic algorithm improves the cost. The following results are obtained from the comparative analysis in the computation time for $n = 25$, and 50 considering ROT(MILP) = 0.01 and 0.1.

- Our heuristic algorithm improves the computation time for medium sized test problems, by 135% and 143% on the NLP-Single start and the NLP-Multi start, respectively (where the optimality gap is 0.01).
- Our heuristic algorithm improves the computation time for large sized test problems, by 232% and 188% on the NLP-Single start and the NLP-Multi start, respectively (where the optimality gap is 0.1).

Table 5.31: The improvement in the average cost computation times by our heuristic algorithm compared with the NLP-Single start

n	10	25			50		
ROT(MILP) value	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of instances that are solved by the NLP-Single start	30	20	30	30	12	24	30
Minimum cost%	2	32	34	32	20	25	40
Computation times%	-11128	142.96	-34.09	-212	232	-191	-621

Table 5.32: The improvement in the average cost and computation times by our heuristic algorithm compared with the NLP-Multi start

n	10	25			50		
ROT(MILP) value	10^{-13}	0.01	0.03	0.05	0.1	0.15	0.2
Number of instances that are solved by the NLP-Multi start	30	20	30	29	10	18	30
Minimum cost%	3.57	30	33	32	29	23	29
Computation times%	-11929.60	135	-54	-515	188	-265	-356

Chapter 6

Conclusion

In this thesis, we made several contributions to the literature of gas distribution network design. The research project aims to develop effective mathematical models for the design of gas distribution networks and to present solution strategies. Considering the underlying graph, there are two different design structures for gas distribution networks normally a tree or a cycle (loop). The structure is motivated by the cost consideration. The tree structure is mostly a cheaper choice of design of the network because all nodes are connected with fewer links than the cycle structure. Therefore, we considered the case of a tree structure. More specifically we contribute the following:

- **The First Contribution:** We developed an effective MINLP model to design the tree network structure and determine its components. These components includes the multi diameters for each link, the pressure at each node, and the flow through each link (Network design and Allocation problem). The objective is to meet demand and requirements at minimum total cost. We consider a wide range of design parameters including the number of demand nodes, the set of pipe diameter types, the length (distance) between nodes, and the pressure limits at each node. We consider the constraints under steady-state conditions such as pressure limits at each node, the flow balance through each link, the pressure drop equation for two ends of each link. The decision variables are the selected links connecting nodes, the flow through each link, the pressure at each node, and the length of selected diameter for each link.

- **The Second Contribution:** We solved our MINLP model using an ap-

proximation method. We chose the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method. The advantage of the OA/ER/AP algorithm over other MINLP algorithms is its ability in handling the nonconvexity of the problem. This algorithm alternates between two sub-problems including a Non Linear Programming (NLP) sub-problem and the Master Mixed Integer Linear Programming (MILP) sub-problem. In the NLP sub-problem, the integer variables are fixed and the continuous variables are determined. In the master MILP sub-problem, the effect of nonconvexities are reduced by linearization of the nonlinear functions at the set of linearization points. Our contribution is that we review this algorithm and then examine its functionality and efficiency on our MINLP model. We consider the effects of different parameters and stopping criteria in the NLP sub-problem and the Master MILP sub-problem to find a near optimal solution. The main outcome is a MINLP model that is effective for small networks.

- **The Third Contribution:** We developed, implemented and tested a new heuristic algorithm to solve our model. We designed the near optimal tree layout and determined its components (multi diameter for each link, the pressure at each node, and flow through each link). Our motivation is to develop a heuristic algorithm to solve larger size problems within 4 hours. We also investigated the quantitative analysis of the effects of different parameters on optimal decisions. We compared the results in the objective function value and the computation time between the OA/ER/AP algorithm and our heuristic algorithm. The main outcome is a heuristic algorithm that finds a cost-effective quality feasible solution for the large sized test problems within 4 hours.

In Chapter 1, we presented an overview of the natural gas including the composition, the formation, and the uses of natural gas as well as the gas supply chain. The gas supply chain is the process of moving the natural gas from the gas field to the consumers. The problems arising in four stages namely production, transmission, distribution, and marketing of gas supply chain have been discussed. In particular, we focused on the gas distribution networks.

In Chapter 2, we presented the literature review for both gas and water dis-

tribution networks. The main problems in gas or water networks are designing optimal structures. In particular, this requires determining the diameter of each pipe in the network. We summarise these problems in more detail below:

- **Fundamental Problem (Problem 1)**

  Network Design and Allocation Problem: Given a set of nodes and customer requirements, determine the optimal pipeline design of the network. This involves determining the

  - Network layout,
  - Diameter of the connected links,
  - Pressure at each node,
  - Flow through each link.

  The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which are tree and cycle (loop).

  There are only a few papers on this fundamental problem. Most of the literature on gas distribution networks has focused on the following sub-problem.

- **Sub-problem (Problem 2)**

  Allocation Problem: Given a set of nodes, set of arcs, network structure and customers requirements, determine the optimal pipeline design of the network. This involves determining the

  - Pipeline diameter for each link,
  - Pressure at each node,
  - Flow through each link.

  The objective usually is to meet demand and requirements at minimum total cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which is a tree or a cycle (loop).

There are three cases of pipeline diameters to consider (Shiono and Suzuki (2016)):

- The Single Diameter (SD), where a link consists of one commercially available diameter.
- The Multi Diameter (MD), where a link consists of different commercially available diameter segments that are serially connected.
- The Continuous Diameter (CD), where a link consists of one diameter type that should be determined between the given minimum and maximum values.

There are few papers in the literature that consider the fundamental problem in cases of single and continuous diameters. There is no work that considers the case of multi diameter. In Chapter 3, we presented an effective MINLP model for Network Design and Allocation Problem in the case of multi diameter. The objective is to meet demand and requirements at minimum cost. The requirements are related to the available pipeline diameters, pressure limits, demands, physics gas laws and network structure which in our case is a tree.

We are motivated to solve our MINLP model using a method that handles the nonconvexities of constraints. The algorithm that we use is the Outer Approximation Algorithm with Equality Relaxation and Augmented Penalty (OA/ER/AP) method given by Viswanathan and Grossmann (1990). Our motivation to choose this OA/ER/AP method among the other MINLP algorithms, is in its ability to handle the nonconvexity of the problem. Our contribution is that we review this algorithm and then examine its functionality and efficiency on our MINLP model. We consider the effects of different parameters and stopping criteria in the levels of the NLP sub-problem and the Master MILP sub-problem to find a near optimal solution. To implement this algorithm to our MINLP model, we partitioned the variables into two subsets of variables: the continuous and the binary variables. Also, we developed the Master MILP sub-problem for our model. We detailed how the equalities and nonconvexities of the constraints are handled in the level of the Master MILP (M-ER/AP) problem. The algorithm terminates if the objective of the Master MILP problem becomes larger than the objective of the NLP problem (in a case of minimisation) or the Master MILP sub-problem is infeasible within the 24 hours time limit. Our main outcome is a MINLP model that is effective for small networks. The details are given in Chapter 3.

In Chapter 4, we test our model using the OA/ER/AP algorithm presented

in Chapter 3. We considered the effective choice of parameters and stopping criteria for the OA/ER/AP algorithm that facilitates the finding of a cost-effective, feasible solution within 24 hours. The number of starting points in the NLP sub-problem and the value of the optimality gap in the Master MILP sub-problem are the main inputs that facilitates the feasibility and efficiency of the solution for our model. A single start implementation might be trapped in a local optimum, but the multiple starts provide a better chance of finding a cost-effective feasible solution. We built our model in commercial software package AIMMS since this algorithm has been implemented in the commercial optimisation package AIMMS named AIMMS Outer Approximation (AOA) algorithm. We generate different sized networks (5, 10, 15, 25 and 50 nodes) to test our model (30 instances for each network size). In this chapter, the OA/ER/AP algorithm is applied to our test cases and we give the following details:

- The computational results with the Single start for the NLP sub-problems. For the medium and large sized problems, we also considered different values of Relative Optimality Tolerance (ROT) for the Master MILP sub-problem.
- The computational results with the Multi start for NLP sub-problems. For the medium and large sized problem, we also considered different values of (ROT) for the Master MLIP sub-problem.
- Comparative analysis between the NLP-Single start and the NLP-Multi start strategies.

From the computational results using the OA/ER/AP algorithm, it turns out that the smaller sized networks are solved when the optimality gap is 10^{-13}. We consider a greater value than 10^{-13} for the optimality gap (the value of ROT(MILP)) for networks with 25 and 50 nodes. We test all 30 instances for specific values of ROT(MILP) under the NLP-Single start and the NLP-Multi start conditions to find a cost-effective feasible solution within 24 hours. As expected the NLP-Multi start yields an improved objective function value but requires more computational time than the NLP-Single start. Our main outcome is a MINLP model that is effective for small networks.

In Chapter 5, we developed a new heuristic algorithm for solving our MINLP model. Our model is computationally difficult to be solved using exact methods in particular for large sized test problems (50 customers). Our motivation is to

develop a heuristic algorithm to solve these large sizes within 4 hours. Our solution strategy is to reduce the level of difficulty by converting the MINLP problem to a Linear Programming problem generating the integer variables in the outer level separately. The developed algorithm includes two levels: in the outer level, a tree network was generated, and in the inner level, we determined the multi diameters for the given links as well as determining the pressure at each node and the flow through each selected link. This method was implemented in JAVA as a programming language, AIMMS as a commercial software package, and Excel as a computer package with functions. The quantitative analysis of the effects of different parameters and stopping criteria of our heuristic algorithm on optimal decisions was also investigated. This chapter included computational results for our algorithm. We also presented a comparative analysis of the objective function value and the computation time between our heuristic algorithm and the OA/ER/AP algorithm. Our main outcome is a heuristic algorithm that finds a cost-effective quality feasible solution for the large sized test problems within 4 hours. The results obtained from the comparative analysis of the objective function value are:

- Our heuristic algorithm improves the objective function value for small size test problems, by 2% and 3.57% on the NLP-Single start and NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for medium size test problems, by 33% and 28% on the NLP-Single start and NLP-Multi start, respectively.
- Our heuristic algorithm improves the average objective function value for large size test problems, by 32% and 27% on the NLP-Single start and NLP-Multi start, respectively.

We also note that an average computation time (sec) with our heuristic algorithm for networks with 10, 25, and 50 nodes of 765.33, 4,351.46, and 10,162.53, respectively. The comparative analysis in the computation time for the small-sized test problems showed that the OA/ER/AP algorithm finds a feasible solution in a shorter computation time than our heuristic algorithm. Also, for networks with 25 and 50 nodes, our heuristic algorithm improves the cost. The following results are obtained from the comparative analysis in the computation time for networks with 25 and 50 nodes:

- Our heuristic algorithm improves the computation time for medium sized test problems, by 135% and 143% on the NLP-Single start and the NLP-Multi start, respectively (when the optimality gap is 0.01).
- Our heuristic algorithm improves the computation time for large sized test problems, by 232% and 188% on the NLP-Single start and the NLP-Multi start, respectively (when the optimality gap is 0.1).

The outcome of the research provided in this thesis given us an effective mathematical model in the area of gas distribution network design and allocation that have unresolved issues. We developed two different solution methods for our model. One is an approximation method and the other a heuristic method. However, there are possible future work in areas that are open to further investigations and perhaps improvements.

6.1 Future Works

Our main goal of this thesis was to consider the network design and allocation problem for natural gas distribution networks. The given set of nodes in our model includes a single source node and demand nodes. Further exploration will be considering more than one source nodes and test our model for networks greater than 50 nodes. Another investigation is to find the impact of demand uncertainty on the model.

There are three cases of pipeline diameters to consider: single diameter; continuous diameter; and multi diameter. In the literature, the Δ-change algorithm has been presented to solve the network design and allocation problem in the cases of single diameter and continuous diameter. In this thesis, we developed an effective Mixed Integer Non Linear Programming (MINLP) model for the fundamental problem in the case of multi diameter. The first investigation is to develop mathematical models for the fundamental problem in the case of single diameter and continuous diameter. This will allow us to compare the performance of the three models for a specific data set. The second investigation is to test these models using our heuristic algorithm. Further exploration will be presenting the comparative analysis between our heuristic algorithm and the Δ-change algo-

rithm.

Another investigation is to speed up our heuristic algorithm. Also, our heuristic algorithm may be utilized to other networks such as water, and electricity networks.

A further different investigation is to develop mathematical models and solution methods for the fundamental problem in the case of cycle structure considering three cases of pipeline diameters.

Appendix

JAVA Code Experts and Permission Statements

```
1  import com.aimms.aimmssdk.AIMMS;
2  import com.aimms.aimmssdk.IConfig;
3  import com.aimms.aimmssdk.IIterator;
4  import com.aimms.aimmssdk.IMultiDimData;
5  import com.aimms.aimmssdk.IProcedure;
6  import com.aimms.aimmssdk.IScalarData;
7  import com.aimms.aimmssdk.ISession;
8  import com.aimms.aimmssdk.ISetData;
9  import com.aimms.aimmssdk.Tuple;
10 import java.io.File;
11 import java.io.IOException;
12 import java.util.ArrayList;
13 import java.util.Arrays;
14 import java.util.Collections;
15 import java.util.Random;
16 import jxl.Cell;
17 import jxl.Sheet;
18 import jxl.Workbook;
19 import jxl.write.Label;
20 import jxl.write.WritableSheet;
21 import jxl.write.WritableWorkbook;
22
23
24 public class Main {
25
26 /**
27 * @param args the command line arguments
28 */
29
30  static int[][] Aopt;
31  static float[][] P;
32  static float[][] R ;
```

```
33  static double [][] C;
34  static ArrayList<int[][]> trees;
35  static ArrayList<double[][]> distance;
36  static ArrayList<Result> U;
37  static ArrayList<Result> results;
38  static ISession session ;
39  static double[][] Min;
40  static double[][] Max;
41
42 /**
43 * We have to initialise these attributes
44 */
45  static int M = ;
46  static int M2 = ;
47  static int Me2 =;
48  static int Me =;
49  static int n = ;
50  static int D = ;
51  static int large = ;
52  static int  small = ;
53  static int SupplyConnection = ;
54  static float alpha = 0.9f;
55  static float epsilon = 0.01f;
56  static int diameterCount = 14;
57  static int r = 5;
58  static int capitalM = 50000;
59  static String aimmsDirectory = "C:\\;
60  static String aimmsProjectDirectory = "C:\\;
61
62
63
64  public static void main(String[] args) {
65
66   myMain();
67
68  }
69
70  public static void myMain(){
71
72   U = new ArrayList<>();
73   Result r = new Result();
74   r.minCost = Double.MAX_VALUE;
75   U.add(r);
```

```
76  R = new float[n][n];
77  C = new double [n][n];
78  int t=0;
79  int[][] A;
80
81  trees = new ArrayList<>();
82
83  /* Reading data from Excel file ------------ */
84  String filePath = "input.xls";
85  String[] nodeSets = new String[n];
86  int[] supply = new int[n];
87  double[][] distance = new double[n][n];
88  String[] diameterTypes = new String[diameterCount];
89  int[] diameters = new int[diameterCount];
90  int[] diameterCosts = new int[diameterCount];
91
92  readExcel(filePath, nodeSets, supply, distance, diameterTypes,
     diameters, diameterCosts);
93  for(String s: nodeSets){
94  System.out.println(s);
95  }
96  for(int i: supply){
97  System.out.println(i);
98  }
99  for(int i=0;i<n ; i++){
100     for(int j=0; j<n ; j++){
101 System.out.print(distance[i][j]+" ");
102     }
103 System.out.println();
104 }
105
106 for(String s: diameterTypes){
107 System.out.println(s);
108 }
109 for(int i: diameters){
110 System.out.println(i);
111 }
112 for(int i: diameterCosts){
113 System.out.println(i);
114 }
115/* ---------------------------------------- */
116
117 copyMatrix1(C, distance);
```

```
118  initializeProbabilityMartix(n,C,large,small);
119  normalizeP( P ,  n );
120  System.out.println("distance is:");
121  for(int i=0; i<n ; i++){
122    for(int j=0; j<n ; j++){
123  System.out.print(distance[i][j]+"  ");
124    }
125  System.out.println();
126  }
127
128
129 while(true){
130  Result result =null;
131  results = new ArrayList<>();
132
133  for(int l=1;l <= M; l++){
134
135  copyMatrix(R, P);
136  System.out.println("ProbMatrix is:");
137  for(int i=0; i<n ; i++){
138    for(int j=0; j<n ; j++){
139  System.out.print(R[i][j]+"  ");
140    }
141  System.out.println();
142  }
143
144  //Here we must specify the number of nodes that we want to create.
145
146  ArrayList<Node> nodes = makeNodes(n);
147  A= buildTree(R, n, nodes);
148
149  boolean repeat = false;
150
151  do{
152  repeat = isTreeRepeated(A);
153  if(repeat){
154  continue;
155  }
156  double x = callAimms1(nodeSets, supply, distance, diameterTypes,
      diameters, diameterCosts, A);
157  if ( x == 0){
158  repeat = true;
159  l-=1;
```

```
160   break;
161   }
162   else{
163   repeat= false;
164   result = new Result();
165   result.A = A;
166   result.minCost = x;
167
168   System.out.println("result for t="+t+" and l="+l+" is : " + result
        .minCost);
169   System.out.println("With tree: ");
170   for(int i=0; i<n ; i++){
171     for(int j=0 ; j<n ; j++){
172   System.out.print(result.A[i][j] + " ");
173     }
174   System.out.println();
175   }
176
177     }
178   }
179   while(repeat);
180
181   results.add(result);
182   trees.add(A);
183
184 }
185
186   sortResults(results);
187   Result c1 = results.get(0);
188   System.out.println("best result for this iteration is: "+c1.
        minCost);
189   if(c1.minCost < U.get(t).minCost){
190  U.add(c1);
191
192   }else{
193  U.add(U.get(t));
194   }
195   boolean stop = false;
196
197   if(checkStopCriteria1(t)){
198   System.out.println("Stop Criteria 1 happend");
199   stop = true;
200   }
```

```
201   if(checkStopCriteria2(P)){
202   System.out.println("Stop Criteria 2 happend");
203   stop = true;
204
205   }
206   if(stop){
207   break;
208   }
209   float[][] V = findRepetitionMatrix(results);
210   P = matrixSum(matrixMultiply((1-alpha), P, n),matrixMultiply(alpha
        , V, n) , n) ;
211
212   for(int k=1 ; k<n ; k++){
213   normalize(P, n, k);
214   }
215   t++;
216
217   if(t>0){
218  M = M2;
219   Me = Me2;
220
221   }
222  }
223
224   System.out.println("Final best result is: " + U.get(t+1).minCost);
225   System.out.println("For tree: ");
226   for(int i=0; i<n ; i++){
227   for(int j=0 ; j<n ; j++){
228   System.out.print(U.get(t+1).A[i][j] + " ");
229   }
230   System.out.println();
231   }
232
233   callAimms2(nodeSets, supply, distance, diameterTypes, diameters,
        diameterCosts, U.get(t+1).A);
234   writeExcel2( t, M, Me, alpha, epsilon);
235   }
236
237 /**
238 * A method to initialise of nodes.
239 * @return
240 */
241 private static ArrayList<Node> makeNodes(int numberOfNodes){
```

```
242   ArrayList<Node> nodes = new ArrayList<>();
243   Node node;
244   for(int i=0 ; i<n ; i++){
245   node = new Node(i);
246   nodes.add(node);
247   }
248   return nodes;
249 }
250 
251 /**
252 * This methods builds a tree.
253 * @param R
254 * @param n
255 * @param nodes
256 * @return
257 */
258 
259 public static int[][] buildTree(float[][] R, int n, ArrayList<Node>
      nodes){
260   int[][] A = new int[n][n];
261   for(int j=1;j < n ; j++){
262       ArrayList<Node> allChilds = nodes.get(j).getAllChilds();
263       for(Node node: allChilds){
264       R[node.id][j] = 0f;
265       }
266     normalize(R,n,j);
267     CDF(R,n,j);
268     double random = Math.random();
269     if(random<R[0][j]){
270     A[0][j]=1;
271     nodes.get(0).addChild(nodes.get(j));
272     }else{
273       for(int i=1 ; i<n ; i++){
274         if(random>R[i-1][j] && random <= R[i][j]){
275         A[i][j]=1;
276         nodes.get(i).addChild(nodes.get(j));
277         break;
278         }
279      }
280     }
281   }
282 
283 return A;
```

```
284
285 }
286
287 /**
288 * A method to normalize column j of matrix.
289 * @param R
290 * @param n
291 * @param j
292 */
293
294 private static void normalize(float[][] R , int n , int j){
295   float sum = 0f;
296   for(int i=0 ; i<n ; i++){
297   sum += R[i][j];
298   }
299   for(int i=0 ; i<n ; i++){
300   R[i][j] = R[i][j] / sum;
301   }
302 }
303
304 /**
305 *
306 * A method to calculate CDF of a column j of matrix.
307 *
308 * @param R
309 * @param n
310 * @param j
311 */
312 private static void CDF(float[][] R, int n, int j){
313   for(int i=1 ; i<n ; i++){
314   R[i][j] = R[i][j] + R[i-1][j];
315   }
316 }
317
318 /**
319 * This is a method to initialize P matrix for the first time.
320 * @param n
321 */
322 private static void initializeProbabilityMartix(int n,double[][] C,
      int large, int small){
323   P = new float[n][n];
324   for (int j=0; j<n; j++){
325     for(int i=0;i<n;i++){
```

```
326       if(i==j  ||  j==0){
327       P[i][j]=0;
328       }else{
329   P[i][j] = (float)(1f/(10*n));
330       }
331       }
332    }
333
334   int t = 0;
335   int z = 0;
336   int w = 0;
337   double max1 = 0;
338   double min = 5000;
339   double min1 = 5000;
340
341   for(int j=1;j<n;j++){
342
343   min = 5000;
344   for(int k =1; k<small; k++){
345     for (int i=0; i<n; i++){
346      if(min> C[i][j]&& i!=j&& C[i][j]!=0){
347        min =C[i][j];
348            z= i;
349        }
350       }
351
352   if( k==1){ P[z][j] = (float)(49f/((n)));
353   }
354   else{P[z][j] = (float)(49f/((n)));
355
356   }
357   C[z][j]=0;
358   min = 5000;
359    }
360     }
361
362
363   for(int j=1;j<n;j++){
364   max1=0;
365   for(int k =1; k<large; k++){
366     for (int i=1; i<n; i++){
367       if(max1< C[i][j]&& i!=j){
368   max1 = C[i][j];
```

```
369   t= i;
370
371       }
372     }
373   C[t][j]=0;
374   P[t][j] = 0f;
375   max1=0;
376     }
377     }
378 }
379
380 private static void normalizeP(float[][] P , int n ){
381
382   for(int j=1 ; j<n ; j++){
383   float sum = 0f;
384   for(int i=0 ; i<n ; i++){
385   sum += P[i][j];
386   }
387   for(int i=0 ; i<n ; i++){
388   P[i][j] = P[i][j] / sum;
389   }
390   }
391 }
392
393 /**
394 * This is a simple method to copy one matrix into another.
395 * @param to
396 * @param from
397 */
398 private static void copyMatrix(float[][] to, float[][] from){
399   for(int i=0;i<n;i++){
400     for(int j=0;j<n;j++){
401       to[i][j] = from[i][j];
402     }
403     }
404 }
405
406 private static void copyMatrix1(double[][] to, double[][] from){
407   for(int i=0;i<n;i++){
408      for(int j=0;j<n;j++){
409       to[i][j] = from[i][j];
410     }
411     }
```

```
412 }
413
414 /**
415 * This method takes array of trees and a tree and checks if this
      tree exists in that
416 * array.
417 *
418 * @param A1
419 * @return
420 */
421 private static boolean isTreeRepeated(int[][][] A1){
422 boolean result = false;
423   for(int[][][] A2:trees){
424     if(isMatrixEqual(A1, A2)){
425       result =  true;
426       break;
427     }
428   }
429    return false;
430 }
431
432 /**
433 * Simple method to check if two matrixes are equal.
434 * @param A1
435 * @param A2
436 * @return
437 */
438 private static boolean isMatrixEqual(int[][][] A1,int[][][] A2){
439   for(int i=0;i<n;i++){
440      for(int j=0;j<n;j++){
441       if(A1[i][j] != A2[i][j]){
442         return false;
443       }
444      }
445   }
446   return true;
447 }
448
449 private static void openAIMMSSession(){
450   session = AIMMS.openSession(aimmsDirectory,aimmsProjectDirectory);
451 }
452
453 private static void closeAIMMSSession(){
```

```
454   session.close();
455 }
456
457
458 /**
459 *
460 * This method gets some arrays that we have gotten from excel and R
      array
461 * This method call AIMMS to get result.
462 *
463 * @param nodeSets
464 * @param supply
465 * @param nodeDistance
466 * @param diameterTypes
467 * @param diameters
468 * @param diameterCosts
469 * @param A
470 */
471 private static double callAimms1(String[] nodeSets , int[] supply ,
      double[][] nodeDistance , String[] diameterTypes , int[]
      diameters , int[] diameterCosts, int[][] A){
472
473
474 openAIMMSSession();
475 IScalarData parameterM = session.openScalar("M");
476 parameterM.setValue(capitalM);
477 parameterM.close();
478
479
480 IMultiDimData parameterSupply = session.openMultiDim("Supply");
481   for(int i=0;i<supply.length ; i++){
482   parameterSupply.insert(new Tuple(nodeSets[i]), supply[i]);
483   }
484 parameterSupply.close();
485
486 IMultiDimData parameterDiameter = session.openMultiDim("Diameter");
487   for(int i=0 ; i<diameterTypes.length ; i++){
488   parameterDiameter.insert(new Tuple(diameterTypes[i]),diameters[i])
      ;
489   }
490 parameterDiameter.close();
491
```

```
IMultiDimData parameterDiameterCost = session.openMultiDim("
    PipeDiameterCost");
  for(int i=0 ; i< diameterTypes.length ; i++){
  parameterDiameterCost.insert(new Tuple(diameterTypes[i]),
    diameterCosts[i]);
  }

parameterDiameterCost.close();
IMultiDimData parameterDistance = session.openMultiDim("Distance");
  for(int i=0 ; i<n ; i++){
     for(int j=0 ; j<n ; j++){
       if(nodeDistance[i][j] != 0){
  parameterDistance.setValue(new Tuple(nodeSets[i],nodeSets[j]),
    nodeDistance[i][j]);
       }
     }
  }
parameterDistance.close();

IMultiDimData parameterArcMap = session.openMultiDim("ArcMap");
  for(int i=0 ; i<n ; i++){
     for(int j=0 ; j<n ; j++){
       if(A[i][j] == 1){
      parameterArcMap.setValue(new Tuple(nodeSets[i],nodeSets[j]),A[
    i][j]);
       }
      }
  }
parameterDistance.close();

IProcedure procedure = session.openProcedure("MainExecution");
procedure.run();
procedure.close();

IScalarData variableTotalCost = session.openScalar("totalcost");
double totalCost = variableTotalCost.asDouble();
variableTotalCost.close();
closeAIMMSSession();
return totalCost;

}
```

```
530 private static void callAimms2(String[] nodeSets , int[] supply ,
    double[][] nodeDistance , String[] diameterTypes , int[]
    diameters , int[] diameterCosts, int[][] A){
531
532 openAIMMSSession();
533 IScalarData parameterM = session.openScalar("M");
534 parameterM.setValue(capitalM);
535 parameterM.close();
536
537 IMultiDimData parameterSupply = session.openMultiDim("Supply");
538   for(int i=0;i<supply.length ; i++){
539   parameterSupply.insert(new Tuple(nodeSets[i]), supply[i]);
540   }
541 parameterSupply.close();
542
543 IMultiDimData parameterDiameter = session.openMultiDim("Diameter");
544   for(int i=0 ; i<diameterTypes.length ; i++){
545   parameterDiameter.insert(new Tuple(diameterTypes[i]),diameters[i])
      ;
546   }
547 parameterDiameter.close();
548
549 IMultiDimData parameterDiameterCost = session.openMultiDim("
    PipeDiameterCost");
550   for(int i=0 ; i< diameterTypes.length ; i++){
551   parameterDiameterCost.insert(new Tuple(diameterTypes[i]),
    diameterCosts[i]);
552   }
553 parameterDiameterCost.close();
554
555 IMultiDimData parameterDistance = session.openMultiDim("Distance");
556   for(int i=0 ; i<n ; i++){
557      for(int j=0 ; j<n ; j++){
558   parameterDistance.setValue(new Tuple(nodeSets[i],nodeSets[j]),
    nodeDistance[i][j]);
559    }
560   }
561 parameterDistance.close();
562
563 IMultiDimData parameterArcMap = session.openMultiDim("ArcMap");
564   for(int i=0 ; i<n ; i++){
565        for(int j=0 ; j<n ; j++){
566          if(A[i][j] == 1){
```

```
567   parameterArcMap.setValue(new Tuple(nodeSets[i],nodeSets[j]),A[i][j
      ]);
568           }
569         }
570   }
571 parameterDistance.close();
572
573 IProcedure procedure = session.openProcedure("MainExecution");
574 procedure.run();
575 procedure.close();
576
577 IScalarData variableTotalCost = session.openScalar("totalcost");
578 double totalCost = variableTotalCost.asDouble();
579 System.out.println();
580 System.out.println();
581 System.out.println("Total Cost: "+totalCost);
582 IMultiDimData variablePresuure = session.openMultiDim("Pressure");
583 IIterator it = variablePresuure.createIterator();
584 double[] pressure = new double[n];
585 int pressureIndex=0;
586 while(it.next()){
587 pressure[pressureIndex] = it.asDouble();
588 pressureIndex++;
589
590 }
591 it.close();
592 //System.out.println("Flow: ");
593 IMultiDimData variableFlow = session.openMultiDim("Flow1");
594 IIterator it3 = variableFlow.createIterator();
595 double[][] flow = new double[n][n];
596   for(int i=0 ; i<n ; i++){
597         for(int j=0 ; j<n ; j++){
598          flow[i][j] = 0;
599         }
600   }
601
602 while(it3.next()){
603   String firstElement = it3.tuple().getElement(0).toString();
604   String secondElement = it3.tuple().getElement(1).toString();
605   Double doubleNumber = it3.asDouble();
606   int column=0;
607   int row=0;
608   for(int i=0; i<n ; i++){
```

```
609     if(nodeSets[i].equals(firstElement.replaceAll("'", "")))){
610      row =i  ;
611   break;
612     }
613   }
614   for(int j=0; j<n  ; j++){
615     if(nodeSets[j].equals(secondElement.replaceAll("'", "")))){
616     column =j  ;
617     break;
618     }
619   }
620   flow[row][column] = doubleNumber;
621 }
622
623   for(int i=0  ; i<n  ; i++){
624     for(int j=0  ; j<n  ; j++){
625     }
626   }
627 it3.close();
628
629
630 IMultiDimData variablePipeDiameterType = session.openMultiDim("
      pipediametertype");
631 IIterator it2 = variablePipeDiameterType.createIterator();
632
633 writeExcel(flow, pressure, it2, totalCost, nodeSets);
634
635 variableTotalCost.close();
636
637 closeAIMMSSession();
638 }
639
640 private static boolean isFeasibleSolution(double h){
641   boolean feasible;
642   if (h == 0){
643   feasible = false;}
644   else{
645   feasible = true;
646   }
647   return feasible;
648 }
649 /**
650 *
```

```
651 * This method sorts an array of Result object.
652 *
653 * @param results
654 */
655 private static void sortResults(ArrayList<Result> results ){
656 Collections.sort(results);
657 }
658
659 private static boolean checkStopCriteria3(int D){
660   boolean isEqual;
661   isEqual= true;
662   for(int i=0; i<D ; i++){
663     if(results.get(i).minCost != results.get(i+1).minCost){
664       isEqual = false;
665     }
666   }
667
668   if( isEqual){
669   return true;
670   }else{
671   return false;
672   }
673 }
674
675 /**
676 *
677 * This method checks for first stop criteria
678 *
679 * @param t
680 * @return
681 */
682
683 private static boolean checkStopCriteria1(int t){
684   boolean isEqual;
685   if(t>=r){
686   isEqual= true;
687     for(int i=t; i>t-r ; i--){
688       if(U.get(i).minCost != U.get(i-1).minCost){
689           isEqual = false;
690       }
691     }
692   }else{
693     isEqual = false;
```

```
694   }
695   if(t >= r && isEqual){
696   return true;
697   }else{
698   return false;
699   }
700 }
701 
702 private static boolean checkStopCriteria2(float[][] P){
703   if(Math.abs(computeNorm(P)-Math.sqrt(n-1)) < epsilon){
704   return true;
705   }else{
706   return false;
707   }
708 }
709 
710 /**
711 * This method calculates V matrix. V[i][j] is the total number of 1'
      s in row j and column j of all Me answers.
712 * @param results
713 * @return
714 */
715 
716 private static float[][] findRepetitionMatrix(ArrayList<Result>
      results){
717   float[][] V = new float[n][n];
718   for(int i=0 ; i<n ; i++){
719      for(int j=0 ; j<n ; j++){
720       float sum = 0;
721        for(int k=0;k<Me ; k++){
722           if(results.get(k).A[i][j] == 1){
723               sum++;
724            }
725         }
726       sum =(float) sum / Me;
727       V[i][j] = sum;
728      }
729   }
730   return V;
731 }
732 
733 /**
734 * Simple method to multiply an integer into a matrix.
```

```
735 * @param x
736 * @param matrix
737 * @param size
738 * @return
739 */
740 private static float[][] matrixMultiply(float x,float[][] matrix,int
        size){
741   for(int i=0 ; i<size ; i++){
742     for(int j=0 ; j<size ; j++){
743       matrix[i][j] = x*matrix[i][j];
744     }
745   }
746   return matrix;
747 }
748
749 /**
750 *
751 * Simple method to summerize two matrixes
752 *
753 * @param matrix1
754 * @param matrix2
755 * @param size
756 * @return
757 */
758
759 private static float[][] matrixSum(float[][] matrix1,float[][]
      matrix2,int size){
760   for(int i=0 ; i<size ; i++){
761     for(int j=0 ; j<size ; j++){
762       matrix1[i][j] = matrix1[i][j] + matrix2[i][j];
763     }
764   }
765   return matrix1;
766 }
767
768 /**
769 *
770 * This method takes some arrays as input and read form excel into
      them.
771 * Note that we have to change startX and startY variables in this
      method manually.
772 *
773 * @param filePath
```

```
774 * @param nodeSets
775 * @param supply
776 * @param nodeDistance
777 * @param diameterTypes
778 * @param diameters
779 * @param diameterCosts
780 */
781
782 private static void readExcel(String filePath, String[] nodeSets,
        int[] supply, double[][] nodeDistance, String[] diameterTypes,
        int[] diameters, int[] diameterCosts){
783 try{
784   Workbook workbook = Workbook.getWorkbook(new File(filePath));
785   Sheet sheet = workbook.getSheet(0);
786
787   int startX=2;
788   int startY=0;
789
790   for(int i=0; i<n ; i++){
791     for(int j=0 ; j<n+2 ; j++){
792       System.out.println(" i="+i+"  j="+j);
793       Cell cell = sheet.getCell(startY+j,startX+i);
794       if(j==0){
795       nodeSets[i] = cell.getContents();
796       }else if(j==1){
797       supply[i] = Integer.parseInt(cell.getContents());
798       }else{
799       nodeDistance[i][j-2] = Double.parseDouble(cell.getContents());
800       }
801       }
802   }
803
804   startX=105;
805   startY=2;
806
807   for(int i=0; i<diameterCount ; i++){
808     for(int j=0 ; j<3 ; j++){
809     Cell cell = sheet.getCell(startY+j , startX+i);
810       if(j==0){
811       diameterTypes[i] = cell.getContents();
812       }else if(j==1){
813       diameters[i] = Integer.parseInt(cell.getContents());
814       }else if(j==2){
```

```
815         diameterCosts[i] = Integer.parseInt(cell.getContents());
816         }
817       }
818     }
819
820     workbook.close();
821 }catch(Exception e){
822 e.printStackTrace();
823 }
824 }
825
826 private static void writeExcel(double[][] flow, double[] pressure,
      IIterator pipeDiameterTypeIt , double totalCost, String[]
      nodeSets){
827 try{
828     WritableWorkbook workbook = Workbook.createWorkbook(new File("
      output.xls"));
829     WritableSheet sheet = workbook.createSheet("First Sheet", 0);
830     int startX=0;
831     int startY=0;
832
833     for(int i=0; i<n+1; i++){
834       for(int j=0 ; j<n+2; j++){
835         if( i==0 && j==0){
836         sheet.addCell(new Label(j+startY, i+startX,"Flow"));
837         }else if(j==0 && i!=0){
838         sheet.addCell(new Label(j+startY, i+startX,nodeSets[i-1]));
839         }else if(i==0 && j!=0 && j!=n+1){
840         sheet.addCell(new Label(j+startY, i+startX,nodeSets[j-1]));
841         }else if(j==n+1 && i==0){
842         sheet.addCell(new Label(j+startY, i+startX,"Pressure"));
843         }else if(j==n+1 && i!=0){
844         sheet.addCell(new Label(j+startY, i+startX,pressure[i-1]+""));
845         }else{
846         sheet.addCell(new Label(j+startY, i+startX,flow[i-1][j-1]+""))
      ;
847         }
848     }
849
850     }
851     startX += n+5;
852     while(pipeDiameterTypeIt.next()){
```

```
853   sheet.addCell(new Label(startY,startX,pipeDiameterTypeIt.tuple().
       toString()));
854   sheet.addCell(new Label(startY+1,startX,pipeDiameterTypeIt.
       asDouble()+""));
855   startX++;
856   }
857   startX += 2;
858   sheet.addCell(new Label(startY,startX,"totalCost"));
859   sheet.addCell(new Label(startY+1,startX,totalCost+""));
860   workbook.write();
861   workbook.close();
862 }catch(Exception e){
863 e.printStackTrace();
864 }
865 pipeDiameterTypeIt.close();
866
867 }
868
869 private static void writeExcel2( int t, int M, int Me,float alpha ,
870 float epsilon){
871   try{
872   WritableWorkbook workbook1 = Workbook.createWorkbook(new File("
       output2.xls"));
873   WritableSheet sheet1 = workbook1.createSheet("First Sheet", 0);
874   int startX=0;
875   int startY=0;
876   sheet1.addCell(new Label(startY,startX,"t = "));
877   sheet1.addCell(new Label(startY+1,startX,t+""));
878   startX=1;
879   startY=0;
880   sheet1.addCell(new Label(startY,startX,"M = "));
881   sheet1.addCell(new Label(startY+1,startX,M+""));
882   startX=2;
883   startY=0;
884   sheet1.addCell(new Label(startY,startX,"Me = "));
885   sheet1.addCell(new Label(startY+1,startX,Me+""));
886
887   startX=3;
888   startY=0;
889   sheet1.addCell(new Label(startY,startX,"alpha = "));
890   sheet1.addCell(new Label(startY+1,startX,alpha+""));
891   startX=4;
892   startY=0;
```

```
893   sheet1.addCell(new Label(startY,startX,"epsilon = "));
894   sheet1.addCell(new Label(startY+1,startX,epsilon+""));
895   startX=5;
896   startY=0;
897   sheet1.addCell(new Label(startY,startX,"r = "));
898
899   startX=6;
900   startY=0;
901   sheet1.addCell(new Label(startY,startX,"Stop Criteria Number = "))
        ;
902   workbook1.write();
903   workbook1.close();
904   }catch(Exception e){
905   e.printStackTrace();
906   }
907   }
908   private static double computeNorm(float[][] P){
909   double sum =0;
910   for(int i=0 ; i<n ; i++){
911   for(int j=0 ; j<n ; j++){
912   sum += Math.pow(P[i][j],2);
913   }
914   }
915   System.out.println("Sum of prob matrix is"+sum);
916   System.out.println("Norm is"+Math.sqrt(sum));
917   return Math.sqrt(sum);
918   }
919 }
920
921
922 \\ Here, we present node class.
923
924 import java.util.ArrayList;
925
926 public class Node {
927
928   Node parent;
929   int id;
930   ArrayList<Node> childs;
931
932   public Node(int id){
933     childs = new ArrayList<>();
934     this.id = id;
```

```
935   }
936
937   public void addChild(Node node){
938     childs.remove(node);
939     childs.add(node);
940   }
941
942   public ArrayList<Node> getChilds(){
943     return childs;
944   }
945
946   private ArrayList<Node> allChilds = new ArrayList<>();
947   private void traverseTree(Node root){
948     allChilds.addAll(root.childs);
949     for(Node node : root.childs){
950       traverseTree(node);
951     }
952   }
953   public ArrayList<Node> getAllChilds(){
954     traverseTree(this);
955     return allChilds;
956   }
957 }
958
959
960 \\Here we present Result class.
961
962 public class Result implements Comparable<Result>{
963
964 float[][] arcMap;
965 float[][] flow;
966 double minCost;
967 int[][] A;
968
969 @Override
970 public int compareTo(Result o) {
971 return Double.compare(minCost, o.minCost);
972  }
973 }
```

SPRINGER LICENSE
TERMS AND CONDITIONS

Dec 07, 2017

This Agreement between Curtin University -- Efat Fakhar ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4235600960362
License date	Nov 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Springer eBook
Licensed Content Title	A New Algorithm for MINLP Applied to Gas Transport Energy Cost Minimization
Licensed Content Author	Björn Geißler, Antonio Morsi, Lars Schewe
Licensed Content Date	Jan 1, 2013
Type of Use	Thesis/Dissertation
Portion	Figures/tables/illustrations
Number of figures/tables/illustrations	2
Author of this Springer article	No
Order reference number	
Original figure numbers	Fig. 2 A & Fig. 3 A
Title of your thesis / dissertation	Optimisation Techniques for Natural gas Distribution Networks
Expected completion date	Feb 2018
Estimated size(pages)	209
Requestor Location	Curtin University Attn: Curtin University
Billing Type	Invoice
Billing Address	Curtin University Attn: Curtin University
Total	0.00 AUD

Figure 6.1: Permission for Figure 1.2

SPRINGER LICENSE
TERMS AND CONDITIONS

Dec 07, 2017

This Agreement between Curtin University -- Efat Fakhar ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

License Number	4235611164082
License date	Nov 24, 2017
Licensed Content Publisher	Springer
Licensed Content Publication	Annals of Operations Research
Licensed Content Title	A Reduction Technique for Natural Gas Transmission Network Optimization Problems
Licensed Content Author	Roger Z. Ríos-Mercado, Suming Wu, L. Ridgway Scott et al
Licensed Content Date	Jan 1, 2002
Licensed Content Volume	117
Licensed Content Issue	1
Type of Use	Thesis/Dissertation
Portion	Figures/tables/illustrations
Number of figures/tables/illustrations	2
Author of this Springer article	No
Order reference number	
Original figure numbers	Figur 1 and Figure 2
Title of your thesis / dissertation	Optimisation Techniques for Natural gas Distribution Networks
Expected completion date	Feb 2018
Estimated size(pages)	209
Requestor Location	Curtin University Australia Attn: Curtin University
Billing Type	Invoice
Billing Address	Curtin University Attn: Curtin University
Total	0.00 USD

Figure 6.2: Permission for Figures 1.3 and 1.4

ELSEVIER LICENSE
TERMS AND CONDITIONS

Dec 07, 2017

This Agreement between Curtin University -- Efat Fakhar ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

License Number	4235610298509
License date	Nov 24, 2017
Licensed Content Publisher	Elsevier
Licensed Content Publication	Energy
Licensed Content Title	Forecasting of natural gas consumption with artificial neural networks
Licensed Content Author	Jolanta Szoplik
Licensed Content Date	Jun 1, 2015
Licensed Content Volume	85
Licensed Content Issue	n/a
Licensed Content Pages	13
Start Page	208
End Page	220
Type of Use	reuse in a thesis/dissertation
Intended publisher of new work	other
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Format	both print and electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Original figure numbers	Fig. 1
Title of your thesis/dissertation	Optimisation Techniques for Natural gas Distribution Networks
Expected completion date	Feb 2018
Estimated size (number of pages)	209
Requestor Location	Curtin University Australia Attn: Curtin University
Total	0.00 AUD

Figure 6.3: Permission for Figure 1.5

Bibliography

Afshar, M. H. (2007). Evaluation of selection algorithms for simultaneous layout and pipe size optimization of water distribution networks. *Scientia Iranica*, 14(1):23–32.

Afshar, M. H., Akbari, M., and Mariño, M. A. (2005). Simultaneous layout and size optimization of water distribution networks: engineering approach. *Journal of Infrastructure Systems*, 11(4):221–230.

Agency, U. S. E. P. (2014). Water research @online. Avaiable at https://www.epa.gov/water-research/epanet.

Alperovits, E. and Shamir, U. (1977). Design of optimal water distribution systems. *Water resources research*, 13(6):885–900.

André, J. (2010). Optimization of investments in gas networks. Université du Littoral Côte d'Opale. Avaiable at https://tel.archives-ouvertes.fr/tel-00539689.

André, J., Auray, S., Brac, J., De Wolf, D., Maisonnier, G., Ould-Sidi, M.-M., and Simonnet, A. (2013). Design and dimensioning of hydrogen transmission pipeline networks. *European Journal of Operational Research*, 229(1):239–251.

Australian Energy, R. (2009). Chapter 10: Gas distribution @online. Available at http://www.aer.gov.au/node/6313.

Bhaskaran, S. and Salzborn, F. J. (1979). Optimal design of gas pipeline networks. *Journal of the Operational Research Society*, pages 1047–1060.

Biegler, L. T. and Grossmann, I. E. (2004). Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192.

Bisschop, J. and Roelofs, M. (2006). *AIMMS language reference*. Lulu. com. Available at https://aimms.com.

Brimberg, J., Hansen, P., Lin, K.-W., Mladenovic, N., and Breton, M. (2003). An oil pipeline design problem. *Operations Research*, 51(2):228–239.

Brkić, D. (2009). An improvement of hardy cross method applied on looped spatial natural gas distribution networks. *Applied energy*, 86(7):1290–1300.

Cobos-Zaleta, D. and Ríos-Mercado, R. Z. (2002). A minlp model for minimizing fuel consumption on natural gas pipeline networks. In *XI Latin-Ibero-American conference on operations research*, pages 27–31.

Cross, H. (1936). Analysis of flow in networks of conduits or conductors. *University of Illinois. Engineering Experiment Station. Bulletin; no. 286.*

De Corte, A. and Sörensen, K. (2013). Optimisation of gravity-fed water distribution network design: A critical review. *European Journal of Operational Research*, 228(1):1–10.

de Mélo Duarte, H., Goldbarg, E. F. G., and Goldbarg, M. C. (2006). A tabu search algorithm for optimization of gas distribution networks. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 37–48. Springer.

De Wolf, D. and Smeers, Y. (1996). Optimal dimensioning of pipe networks with application to gas transmission networks. *Operations Research*, 44(4):596–608.

De Wolf, D. and Smeers, Y. (2000). The gas transmission problem solved by an extension of the simplex algorithm. *Management Science*, 46(11):1454–1465.

Deb, K. and Agrawal, S. (1999). A niched-penalty approach for constraint handling in genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms*, pages 235–243. Springer.

Demissie, A. and Zhu, W. (2015). A survey on gas pipelines operation and design optimization. In *IIE Annual Conference. Proceedings*, page 734. Institute of Industrial Engineers-Publisher.

Djebedjian, B., Mohamed, M. S., Mondy, A.-G., and Rayan, M. A. (2005). Network optimization for steady flow and water hammer using genetic algorithms. In *Ninth International Water Technology Conference, IWTC 2005*, pages 1101–1115.

Duran, M. A. and Grossmann, I. E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339.

Eshragh, A., Filar, J., and Nazar, A. (2011). A projection-adapted cross entropy (pace) method for transmission network planning. *Energy Systems*, 2(2):189–208.

Eusuff, M. M. and Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management*, 129(3):210–225.

Fletcher, R. and Leyffer, S. (1994). Solving mixed integer nonlinear programs by outer approximation. *Mathematical programming*, 66(1-3):327–349.

Gabriel, S. A., Kiet, S., and Zhuang, J. (2005). A mixed complementarity-based equilibrium model of natural gas markets. *Operations Research*, 53(5):799–818.

Geißler, B., Morsi, A., and Schewe, L. (2013). A new algorithm for minlp applied to gas transport energy cost minimization. In *Facets of Combinatorial Optimization*, pages 321–353. Springer.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260.

Gill, P. E., Murray, W., and Saunders, M. A. (2005). Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131.

Goulter, I. C. (1992). Systems analysis in water-distribution network design: From theory to practice. *Journal of Water Resources Planning and Management*, 118(3):238–248.

Grossmann, I. E. and Kravanja, Z. (1995). Mixed-integer nonlinear programming techniques for process systems engineering. *Computers & Chemical Engineering*, 19:189–204.

Haghighi, A., Samani, H. M., and Samani, Z. M. (2011). Ga-ilp method for optimization of water distribution networks. *Water Resources Management*, 25(7):1791–1808.

Hamedi, M., Farahani, Z., and Esmaeilian, G. (2011). Optimization in natural gas network planning. *Logistics operations and management, 1st edn. Elsevier, London*, pages 393–420.

Kan, A. H. R., Boender, C. G. E., and Timmer, G. T. (1985). A stochastic approach to global optimization. In *Computational mathematical programming*, pages 281–308. Springer.

Kan, A. H. R. and Timmer, G. T. (1987). Stochastic global optimization methods part ii: Multi level methods. *Mathematical Programming*, 39(1):57–78.

Kocis, G. R. and Grossmann, I. E. (1987). Relaxation strategy for the structural optimization of process flow sheets. *Industrial & engineering chemistry research*, 26(9):1869–1880.

Krope, J., Trop, P., and Goricanec, D. (2011). Flow-pressure analysis of loop gas networks. *International journal of systems applications, engineering & development*, 5(4):477–484.

Mardaneh, E. and Caccetta, L. (2016). Adapted cross entropy method to investigate costly price-changes in pricing and production planning. *Pacific Journal of Optimization*, 12(2):399–414.

Mardaneh, E., Lin, Q., and Loxton, R. (2015). A heuristic algorithm for optimal fleet composition with vehicle routing considerations. *Optimization Methods and Software*, pages 1–18.

Melo, W., Fampa, M., and Raupp, F. (2014). Integrating nonlinear branch-and-bound and outer approximation for convex mixed integer nonlinear programming. *Journal of Global Optimization*, 60(2):373–389.

Mohajeri, A., Mahdavi, I., and Mahdavi-Amiri, N. (2012). Optimal pipe diameter sizing in a tree–structured gas network: a case study. *International Journal of Industrial and Systems Engineering*, 12(3):346–368.

Murray III, J. E. and Edgar, T. F. (1978). Optimal scheduling of production and compression in gas fields. *Journal of Petroleum Technology*, 30(01):109–116.

Nasr, G. G. and Connor, N. E. (2014). Natural gas engineering and safety challenges. Available at http://www.springer.com.

NatGas (2013). Overview of natural gas @online. Available at http://naturalgas.org/.

Nikbakht, M., Zulkifli, N., Ismail, N., Sulaiman, S., Sadrnia, A., and Suleiman, M. (2012). Multi-echelon supply chain design in natural gas industry. *World Applied Sciences Journal*, 20(1):54–63.

O'Neill, R. P., Williard, M., Wilkins, B., and Pike, R. (1979). A mathematical programming model for allocation of natural gas. *Operations Research*, 27(5):857–873.

Osiadacs, A. J. and Pienkosz, K. (1988). Methods of steady-state simulation for gas networks. *International Journal of Systems Science*, 19(7):1311–1321.

Parker, N. (2004). Using natural gas transmission pipeline costs to estimate hydrogen pipeline costs. Available at https://EconPapers.repec.org/RePEc:cdl:itsdav:qt9m40m75r.

Quesada, I. and Grossmann, I. E. (1992). An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & chemical engineering*, 16(10-11):937–947.

Ranjbar, B. (2011). Gas station @online. Available at http://search.4shared.com.

Raoni, R., Secchi, A. R., and Biscaia, E. C. (2017). Novel method for looped pipeline network resolution. *Computers & Chemical Engineering*, 96:169–182.

Ríos-Mercado, R. Z. and Borraz-Sánchez, C. (2015). Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, 147:536–555.

Ríos-Mercado, R. Z., Wu, S., Scott, L. R., and Boyd, E. A. (2002). A reduction technique for natural gas transmission network optimization problems. *Annals of Operations Research*, 117(1):217–234.

Roarty, M. J. and Roarty, M. (2008). *Australia's natural gas: issues and trends*. Australia. Department of Parliamentary Services. Parliamentary Library.

Rothfarb, B., Frank, H., Rosenbaum, D. M., Steiglitz, K., and Kleitman, D. J. (1970). Optimal design of offshore natural-gas pipeline systems. *Operations research*, 18(6):992–1020.

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112.

Rubinstein, R. Y. and Kroese, D. P. (2013). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media.

Rubio-Barros, R., Ojeda-Esteybar, D. M., Ano, O., and Vargas, A. (2008). Integrated natural gas and electricity market: A survey of the state of the art in operation planning and market issues. In *Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES*, pages 1–8. IEEE.

Ruiz, C., Conejo, A. J., Fuller, J. D., Gabriel, S. A., and Hobbs, B. F. (2014). A tutorial review of complementarity models for decision-making in energy markets. *EURO Journal on Decision Processes*, 2(1-2):91–120.

Saldarriaga, C. A., Hincapié, R. A., and Salazar, H. (2013). A holistic approach for planning natural gas and electricity distribution networks. *IEEE transactions on power systems*, 28(4):4052–4063.

Saleh, S. H. and Tanyimboh, T. T. (2013). Coupled topology and pipe size optimization of water distribution systems. *Water resources management*, 27(14):4795–4814.

Sanaye, S. and Nasab, A. M. (2012). Modeling and optimizing a chp system for natural gas pressure reduction plant. *Energy*, 40(1):358–369.

Sarbu, I. (2014). Nodal analysis of urban water distribution networks. *Water resources management*, 28(10):3143–3159.

Shiono, N. and Suzuki, H. (2016). Optimal pipe-sizing problem of tree-shaped gas distribution networks. *European Journal of Operational Research*, 252(2):550–560.

Smith, A. E. and Coit, D. W. (1997). Penalty functions. *Handbook on Evolutionary Computation, pages C*, 5:1–6.

Spiliotis, M. and Tsakiris, G. (2013). Closure to "water distribution system analysis: Newton-raphson method revisited" by m. spiliotis and g. tsakiris. *Journal of Hydraulic Engineering*, 139(8):918–919.

Stevens, B. (2012). Barry on energy @online. Available at https://barryonenergy.wordpress.com.

Suribabu, C. (2012). Heuristic-based pipe dimensioning model for water distribution networks. *Journal of Pipeline Systems Engineering and Practice*, 3(4):115–124.

Szoplik, J. (2015). Forecasting of natural gas consumption with artificial neural networks. *Energy*, 85:208–220.

Tabkhi, F., Pibouleau, L., Azzaro-Pantel, C., and Domenech, S. (2009). Total cost minimization of a high-pressure natural gas network. *Journal of Energy Resources Technology*, 131(4):043002.

Ting-zhe, N. (2006). Optimal lay-out of natural gas pipeline network. In *23rd World Gas Conference, Amsterdam*.

Tingzhe, N. and Changgui, D. (2005). Layout optimization of gas transmission system by hopfield neural network. *NATUR. GAS IND*, 2:155–157.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2007). Scatter search and local nlp solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3):328–340.

Uraikul, V., Chan, C., and Tontiwachwuthikul, P. (2004). A mixed-integer optimization model for compressor selection in natural gas pipeline network system operations. *Journal of Environmental Informatics*, 3(1):33–41.

Viswanathan, J. and Grossmann, I. E. (1990). A combined penalty function and outer-approximation method for minlp optimization. *Computers & Chemical Engineering*, 14(7):769–782.

Wu, S., Scott, L. R., Rıos-Mercado, R. Z., and Boyd, E. A. (2000). A network reduction technique for natural gas pipeline networks. *Proceedings of the X CLAIO A*, 281.

Wu, Y., Lai, K. K., and Liu, Y. (2007). Deterministic global optimization approach to steady-state distribution gas pipeline networks. *Optimization and Engineering*, 8(3):259–275.

Zheng, F., Simpson, A. R., Zecchin, A. C., and Deuerlein, J. W. (2013). A graph decomposition-based approach for water distribution network optimization. *Water Resources Research*, 49(4):2093–2109.

Zheng, Q. P., Rebennack, S., Iliadis, N. A., and Pardalos, P. M. (2010). Optimization models in the natural gas industry. In *Handbook of Power Systems I*, pages 121–148. Springer.

Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copy right owner who has been omitted or incorrectly acknowledged.